知道创宇

密级 ★ | 项目组公开

智能合约审计报告

安全状态

安全

★ ★ ★ ★ ★

主测人： 知道创宇区块链安全研究团队

版本说明

修订内容	时间	修订者	版本号
编写文档	20210330	知道创宇区块链安全研究团队	V1.0

文档信息

文档名称	文档版	报告编号	保密级别
YouSwap 智能合约审计报告	V1.0		项目组公开

声明

创宇仅就本报告出具前已经发生或存在的事实出具本报告，并就此承担相应责任。对于出具以后发生或存在的事实，创宇无法判断其智能合约安全状况，亦不对此承担责任。本报告所作的安全审计分析及其他内容，仅基于信息提供者截至本报告出具时向创宇提供的文件和资料。创宇假设:已提供资料不存在缺失、被篡改、删减或隐瞒的情形。如已提供资料信息缺失、被篡改、删减、隐瞒或反映的情况与实际情况不符的，创宇对由此而导致的损失和不利影响不承担任何责任。

目录

1. 综述

本次报告有效测试时间是从 2021 年 3 月 26 日开始到 2021 年 3 月 27 日结束，在此期间针对 **YouSwap 智能合约代码**的安全性和规范性进行审计并以此作为报告统计依据。

此次测试中，知道创宇工程师对智能合约的常见漏洞（见第四章节）进行了全面的分析，同时对业务逻辑层面进行审计，未发现存在相关安全风险以及逻辑设错误，故对此合约综合评定为**通过**。

本次智能合约安全审计结果：通过

由于本次测试过程在非生产环境下进行，所有代码均为最新备份，测试过程均与相关接口人进行沟通，并在操作风险可控的情况下进行相关测试操作，以规避测试过程中的生产运营风险、代码安全风险。

本次审计的报告信息：

报告编号：

报告查询地址链接:

本次审计的目标信息：

条目	描述	
项目名称	Youswap	
合约地址	YouswapFactoryV1	https://etherscan.io/address/0x0604F2781eF5712130Af4d941cb79257513d1693#code
代码类型	以太坊智能合约代码	
代码语言	Solidity	

合约文件及哈希：

合约文件	MD5
YouswapFactoryV1.sol	109624393D08BECD7737278580593182

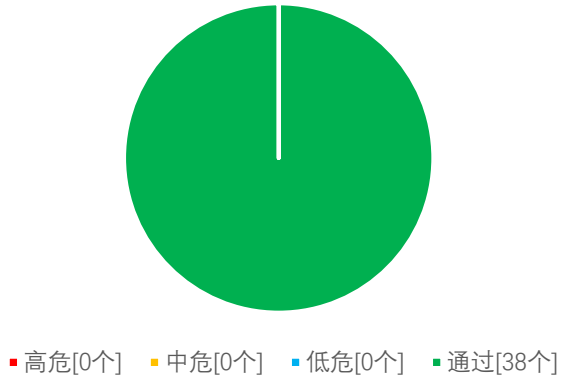
2. 代码漏洞分析

2.1 漏洞等级分布

本次漏洞风险按等级统计：

安全风险等级个数统计表			
高危	中危	低危	通过
0	0	0	38

风险等级分布图



2.2 审计结果汇总说明

审计结果			
审计项目	审计内容	状态	描述
业务安全	邀请处理逻辑设计	通过	经检测，不存在该安全问题。
	用户注册逻辑设计	通过	经检测，不存在该安全问题。
	owner 权限转移	通过	经检测，不存在该安全问题。
	用户质押逻辑设计	通过	经检测，不存在该安全问题。
	奖励提取逻辑设计	通过	经检测，不存在该安全问题。
	算力占比计算逻辑	通过	经检测，不存在该安全问题。
	待领取奖励逻辑设计	通过	经检测，不存在该安全问题。
	下级收益贡献逻辑设计	通过	经检测，不存在该安全问题。
	个人收益加成逻辑	通过	经检测，不存在该安全问题。
	新建矿池逻辑设计	通过	经检测，不存在该安全问题。
	矿池信息修改逻辑	通过	经检测，不存在该安全问题。
编码安全	编译器版本安全	通过	经检测，不存在该安全问题。
	冗余代码	通过	经检测，不存在该安全问题。
	安全算数库的使用	通过	经检测，不存在该安全问题。
	不推荐的编码方式	通过	经检测，不存在该安全问题。
	require/assert 的合理使用	通过	经检测，不存在该安全问题。
	fallback 函数安全	通过	经检测，不存在该安全问题。
	tx.orgin 身份验证	通过	经检测，不存在该安全问题。
	owner 权限控制	通过	经检测，不存在该安全问题。
	gas 消耗检测	通过	经检测，不存在该安全问题。
	call 注入攻击	通过	经检测，不存在该安全问题。
	低级函数安全	通过	经检测，不存在该安全问题。
	增发代币漏洞	通过	经检测，不存在该安全问题。

	访问控制缺陷检测	通过	经检测，不存在该安全问题。
	数值溢出检测	通过	经检测，不存在该安全问题。
	算数精度误差	通过	经检测，不存在该安全问题。
	错误使用随机数检测	通过	经检测，不存在该安全问题。
	不安全的接口使用	通过	经检测，不存在该安全问题。
	变量覆盖	通过	经检测，不存在该安全问题。
	未初始化的存储指针	通过	经检测，不存在该安全问题。
	返回值调用验证	通过	经检测，不存在该安全问题。
	交易顺序依赖	通过	经检测，不存在该安全问题。
	时间戳依赖攻击	通过	经检测，不存在该安全问题。
	拒绝服务攻击	通过	经检测，不存在该安全问题。
	假充值漏洞	通过	经检测，不存在该安全问题。
	重入攻击检测	通过	经检测，不存在该安全问题。
	重放攻击检测	通过	经检测，不存在该安全问题。
	重排攻击检测	通过	经检测，不存在该安全问题。

3. 业务安全性检测

3.1 邀请处理逻辑设计【通过】

对用户邀请处理逻辑设计进行审计，检查是否有对参数进行校验、函数调用者权限校验、以及相关逻辑设计是否合理。

审计结果：相关逻辑设计合理无误。

```
function inviteCount() override external view returns (uint256) {
        return inviteUserInfoV1.length;
    }

    function inviteUpper1(address _owner) override external view returns (address) {
        return inviteUserInfoV2[_owner].upper;
    }

    function inviteUpper2(address _owner) override external view returns (address, address) {
        address upper1 = inviteUserInfoV2[_owner].upper;
        address upper2 = address(0);
        if (address(0) != upper1) {
            upper2 = inviteUserInfoV2[upper1].upper;
        }

        return (upper1, upper2);
    }

    function inviteLower1(address _owner) override external view returns (address[] memory) {
        return inviteUserInfoV2[_owner].lowers;
    }

    function inviteLower2(address _owner) override external view returns (address[] memory,
```

```
address[] memory) {
        address[] memory lowers1 = inviteUserInfoV2[_owner].lowers;
        uint256 count = 0;
        uint256 lowers1Len = lowers1.length;
        for (uint256 i = 0; i < lowers1Len; i++) {
            count += inviteUserInfoV2[lowers1[i]].lowers.length;
        }
        address[] memory lowers;
        address[] memory lowers2 = new address[](count);
        count = 0;
        for (uint256 i = 0; i < lowers1Len; i++) {
            lowers = inviteUserInfoV2[lowers1[i]].lowers;
            for (uint256 j = 0; j < lowers.length; j++) {
                lowers2[count] = lowers[j];
                count++;
            }
        }

        return (lowers1, lowers2);
    }

    function inviteLower2Count(address _owner) override external view returns (uint256, uint256) {
        address[] memory lowers1 = inviteUserInfoV2[_owner].lowers;
        uint256 lowers2Len = 0;
        uint256 len = lowers1.length;
        for (uint256 i = 0; i < len; i++) {
            lowers2Len += inviteUserInfoV2[lowers1[i]].lowers.length;
        }
        return (lowers1.length, lowers2Len);
    }
    function acceptInvitation(address _inviter) override external returns (bool) {
        require(msg.sender != _inviter, ErrorCode.FORBIDDEN);
        UserInfo storage user = inviteUserInfoV2[msg.sender];
```

```
        require(0 == user.startBlock, ErrorCode.REGISTERED);
        UserInfo storage upper = inviteUserInfoV2[_inviter];
        if (0 == upper.startBlock) {
            upper.upper = ZERO;
            upper.startBlock = block.number;
            inviteUserInfoV1.push(_inviter);

            emit InviteV1(_inviter, upper.upper, upper.startBlock);
        }
        user.upper = _inviter;
        upper.lowers.push(msg.sender);
        user.startBlock = block.number;
        inviteUserInfoV1.push(msg.sender);

        emit InviteV1(msg.sender, user.upper, user.startBlock);

        return true;
    }
    function acceptInvitation(address _inviter) override external returns (bool) {
        require(msg.sender != _inviter, ErrorCode.FORBIDDEN);
        UserInfo storage user = inviteUserInfoV2[msg.sender];
        require(0 == user.startBlock, ErrorCode.REGISTERED);
        UserInfo storage upper = inviteUserInfoV2[_inviter];
        if (0 == upper.startBlock) {
            upper.upper = ZERO;
            upper.startBlock = block.number;
            inviteUserInfoV1.push(_inviter);

            emit InviteV1(_inviter, upper.upper, upper.startBlock);
        }
        user.upper = _inviter;
        upper.lowers.push(msg.sender);
        user.startBlock = block.number;
```

```
        inviteUserInfoV1.push(msg.sender);

        emit InviteV1(msg.sender, user.upper, user.startBlock);

        return true;
    }
    function inviteBatch(address[] memory _invitees) override external returns (uint, uint) {
        uint len = _invitees.length;
        require(len <= 100, ErrorCode.PARAMETER_TOO_LONG);
        UserInfo storage user = inviteUserInfoV2[msg.sender];
        if (0 == user.startBlock) {
            user.upper = ZERO;
            user.startBlock = block.number;
            inviteUserInfoV1.push(msg.sender);

            emit InviteV1(msg.sender, user.upper, user.startBlock);
        }
        uint count = 0;
        for (uint i = 0; i < len; i++) {
            if ((address(0) != _invitees[i]) && (msg.sender != _invitees[i])) {
                UserInfo storage lower = inviteUserInfoV2[_invitees[i]];
                if (0 == lower.startBlock) {
                    lower.upper = msg.sender;
                    lower.startBlock = block.number;
                    user.lowers.push(_invitees[i]);
                    inviteUserInfoV1.push(_invitees[i]);
                    count++;

                    emit InviteV1(_invitees[i], msg.sender, lower.startBlock);
                }
            }
        }
```

```
        return (len, count);
    }
```

安全建议：无。

3.2 用户注册逻辑设计【通过】

对用户注册逻辑设计进行审计，检查是否有对参数进行校验、函数调用者权限校验、以及相关逻辑设计是否合理。

审计结果：相关逻辑设计合理无误。

```
    function register() override external returns (bool) {
        UserInfo storage user = inviteUserInfoV2[tx.origin];
        require(0 == user.startBlock, ErrorCode.REGISTERED);
        user.upper = ZERO;
        user.startBlock = block.number;
        inviteUserInfoV1.push(tx.origin);
        emit InviteV1(tx.origin, user.upper, user.startBlock);

        return true;
    }
```

安全建议：无。

3.3 owner 权限转移【通过】

对合约 owner 权限转移逻辑设计进行审计，检查是否有对参数进行校验、函数调用者权限校验、以及相关逻辑设计是否合理。

审计结果：相关逻辑设计合理无误。

```
    function transferOwnership(address _owner) override external {
        require(owner == msg.sender, ErrorCode.FORBIDDEN);
        require((address(0) != _owner) && (owner != _owner), ErrorCode.INVALID_ADDRESSES);
```

```
        _setOperateOwner(owner, false);
        _setOperateOwner(_owner, true);
        owner = _owner;
    }
        function _setOperateOwner(address _address, bool _bool) internal {
        require(owner == msg.sender, ErrorCode.FORBIDDEN);
        operateOwner[_address] = _bool;
    }
```

安全建议：无。

3.4 用户质押逻辑设计【通过】

对质押逻辑设计进行审计，检查是否有对参数进行校验、函数调用者权限校验、以及相关逻辑设计是否合理。

审计结果：相关逻辑设计合理无误。

```
    function deposit(uint256 _pool, uint256 _amount) override external {
        require(0 < _amount, ErrorCode.FORBIDDEN);
        PoolInfo storage poolInfo = poolInfos[_pool];
        require((address(0) != poolInfo.lp) && (poolInfo.startBlock <= block.number),
ErrorCode.MINING_NOT_STARTED);
        //require(0 == poolInfo.endBlock, ErrorCode.END_OF_MINING);
        (, uint256 startBlock) = invite.inviteUserInfoV2(msg.sender);
        if (0 == startBlock) {
            invite.register();
            emit InviteRegister(msg.sender);
        }
        IERC20(poolInfo.lp).safeTransferFrom(msg.sender, address(this), _amount);

        (address upper1, address upper2) = invite.inviteUpper2(msg.sender);
        computeReward(_pool);
        provideReward(_pool, poolInfo.rewardPerShare, poolInfo.lp, msg.sender, upper1, upper2);
```

```
        addPower(_pool, msg.sender, _amount, upper1, upper2);
        setRewardDebt(_pool, poolInfo.rewardPerShare, msg.sender, upper1, upper2);
        emit Stake(_pool, poolInfo.lp, msg.sender, _amount);
    }
    function computeReward(uint256 _pool) internal {
        PoolInfo storage poolInfo = poolInfos[_pool];
        if ((0 < poolInfo.totalPower) && (poolInfo.rewardProvide < poolInfo.rewardTotal)) {
            uint256 reward = (block.number - poolInfo.lastRewardBlock).mul(poolInfo.rewardPerBlock);
            if (poolInfo.rewardProvide.add(reward) > poolInfo.rewardTotal) {
                reward = poolInfo.rewardTotal.sub(poolInfo.rewardProvide);
                poolInfo.endBlock = block.number;
            }

            rewardTotal = rewardTotal.add(reward);
            poolInfo.rewardProvide = poolInfo.rewardProvide.add(reward);
            poolInfo.rewardPerShare = poolInfo.rewardPerShare.add(reward.mul(1e24).div(poolInfo.totalPower));
            poolInfo.lastRewardBlock = block.number;

            emit Mint(_pool, poolInfo.lp, reward);

            if (0 < poolInfo.endBlock) {
                emit EndPool(_pool, poolInfo.lp);
            }
        }
    }
    function provideReward(uint256 _pool, uint256 _rewardPerShare, address _lp, address _user, address _upper1, address _upper2) internal {
        uint256 inviteReward = 0;
        uint256 pledgeReward = 0;
        UserInfo storage userInfo = pledgeUserInfo[_pool][_user];
        if ((0 < userInfo.invitePower) || (0 < userInfo.pledgePower)) {
```

```
                inviteReward =
userInfo.invitePower.mul(_rewardPerShare).sub(userInfo.inviteRewardDebt).div(1e24);
                pledgeReward =
userInfo.pledgePower.mul(_rewardPerShare).sub(userInfo.pledgeRewardDebt).div(1e24);

                userInfo.pendingReward =
userInfo.pendingReward.add(inviteReward.add(pledgeReward));

                RewardInfo storage userRewardInfo = rewardInfos[_user];
                userRewardInfo.inviteReward = userRewardInfo.inviteReward.add(inviteReward);
                userRewardInfo.pledgeReward = userRewardInfo.pledgeReward.add(pledgeReward);
            }

            if (0 < userInfo.pendingReward) {
                you.mint(_user, userInfo.pendingReward);

                RewardInfo storage userRewardInfo = rewardInfos[_user];
                userRewardInfo.receiveReward = userRewardInfo.inviteReward;

                emit WithdrawReward(_pool, _lp, _user, userInfo.pendingReward);

                userInfo.pendingReward = 0;
            }

            if (address(0) != _upper1) {
                UserInfo storage upper1Info = pledgeUserInfo[_pool][_upper1];
                if ((0 < upper1Info.invitePower) || (0 < upper1Info.pledgePower)) {
                    inviteReward =
upper1Info.invitePower.mul(_rewardPerShare).sub(upper1Info.inviteRewardDebt).div(1e24);
                    pledgeReward =
upper1Info.pledgePower.mul(_rewardPerShare).sub(upper1Info.pledgeRewardDebt).div(1e24);

                    upper1Info.pendingReward =
```

```
upper1Info.pendingReward.add(inviteReward.add(pledgeReward));

                RewardInfo storage upper1RewardInfo = rewardInfos[_upper1];
                upper1RewardInfo.inviteReward =
upper1RewardInfo.inviteReward.add(inviteReward);
                upper1RewardInfo.pledgeReward =
upper1RewardInfo.pledgeReward.add(pledgeReward);
            }

            if (address(0) != _upper2) {
                UserInfo storage upper2Info = pledgeUserInfo[_pool][_upper2];
                if ((0 < upper2Info.invitePower) || (0 < upper2Info.pledgePower)) {
                    inviteReward =
upper2Info.invitePower.mul(_rewardPerShare).sub(upper2Info.inviteRewardDebt).div(1e24);
                    pledgeReward =
upper2Info.pledgePower.mul(_rewardPerShare).sub(upper2Info.pledgeRewardDebt).div(1e24);

                    upper2Info.pendingReward =
upper2Info.pendingReward.add(inviteReward.add(pledgeReward));

                    RewardInfo storage upper2RewardInfo = rewardInfos[_upper2];
                    upper2RewardInfo.inviteReward =
upper2RewardInfo.inviteReward.add(inviteReward);
                    upper2RewardInfo.pledgeReward =
upper2RewardInfo.pledgeReward.add(pledgeReward);
                }
            }
        }
    }
    function addPower(uint256 _pool, address _user, uint256 _amount, address _upper1, address
_upper2) internal {
        PoolInfo storage poolInfo = poolInfos[_pool];
        poolInfo.amount = poolInfo.amount.add(_amount);
```

```
uint256 pledgePower = _amount;
UserInfo storage userInfo = pledgeUserInfo[_pool][_user];
userInfo.amount = userInfo.amount.add(_amount);
userInfo.pledgePower = userInfo.pledgePower.add(pledgePower);
poolInfo.totalPower = poolInfo.totalPower.add(pledgePower);
if (0 == userInfo.startBlock) {
    userInfo.startBlock = block.number;
    pledgeAddresss[_pool].push(msg.sender);
}

uint256 upper1InvitePower = 0;
uint256 upper2InvitePower = 0;

if (address(0) != _upper1) {
    uint256 inviteSelfPower = pledgePower.mul(inviteSelfReward).div(100);
    userInfo.invitePower = userInfo.invitePower.add(inviteSelfPower);
    poolInfo.totalPower = poolInfo.totalPower.add(inviteSelfPower);

    uint256 invite1Power = pledgePower.mul(invite1Reward).div(100);
    UserInfo storage upper1Info = pledgeUserInfo[_pool][_upper1];
    upper1Info.invitePower = upper1Info.invitePower.add(invite1Power);
    upper1InvitePower = upper1Info.invitePower;
    poolInfo.totalPower = poolInfo.totalPower.add(invite1Power);
    if (0 == upper1Info.startBlock) {
        upper1Info.startBlock = block.number;
        pledgeAddresss[_pool].push(_upper1);
    }
}

if (address(0) != _upper2) {
    uint256 invite2Power = pledgePower.mul(invite2Reward).div(100);
    UserInfo storage upper2Info = pledgeUserInfo[_pool][_upper2];
```

```
                upper2Info.invitePower = upper2Info.invitePower.add(invite2Power);
                upper2InvitePower = upper2Info.invitePower;
                poolInfo.totalPower = poolInfo.totalPower.add(invite2Power);
                if (0 == upper2Info.startBlock) {
                    upper2Info.startBlock = block.number;
                    pledgeAddresss[_pool].push(_upper2);
                }
            }

            emit UpdatePower(_pool, poolInfo.lp, poolInfo.totalPower, _user, userInfo.invitePower, userInfo.pledgePower, _upper1, upper1InvitePower, _upper2, upper2InvitePower);
        }
        function setRewardDebt(uint256 _pool, uint256 _rewardPerShare, address _user, address _upper1, address _upper2) internal {
            UserInfo storage userInfo = pledgeUserInfo[_pool][_user];
            userInfo.inviteRewardDebt = userInfo.invitePower.mul(_rewardPerShare);
            userInfo.pledgeRewardDebt = userInfo.pledgePower.mul(_rewardPerShare);

            if (address(0) != _upper1) {
                UserInfo storage upper1Info = pledgeUserInfo[_pool][_upper1];
                upper1Info.inviteRewardDebt = upper1Info.invitePower.mul(_rewardPerShare);
                upper1Info.pledgeRewardDebt = upper1Info.pledgePower.mul(_rewardPerShare);

                if (address(0) != _upper2) {
                    UserInfo storage upper2Info = pledgeUserInfo[_pool][_upper2];
                    upper2Info.inviteRewardDebt = upper2Info.invitePower.mul(_rewardPerShare);
                    upper2Info.pledgeRewardDebt = upper2Info.pledgePower.mul(_rewardPerShare);
                }
            }
        }
```

安全建议：无。

3.5 奖励提取逻辑设计【通过】

对提款逻辑设计进行审计，检查是否有对参数进行校验、函数调用者权限校验、以及相关逻辑设计是否合理。

审计结果：相关逻辑设计合理无误。

```
    function withdraw(uint256 _pool, uint256 _amount) override external {
        PoolInfo storage poolInfo = poolInfos[_pool];
        require((address(0) != poolInfo.lp) && (poolInfo.startBlock <= block.number),
ErrorCode.MINING_NOT_STARTED);
        if (0 < _amount) {
            UserInfo storage userInfo = pledgeUserInfo[_pool][msg.sender];
            require(_amount <= userInfo.amount, ErrorCode.BALANCE_INSUFFICIENT);
            IERC20(poolInfo.lp).safeTransfer(msg.sender, _amount);

            emit UnStake(_pool, poolInfo.lp, msg.sender, _amount);
        }
        (address _upper1, address _upper2) = invite.inviteUpper2(msg.sender);
        computeReward(_pool);
        provideReward(_pool, poolInfo.rewardPerShare, poolInfo.lp, msg.sender, _upper1, _upper2);

        if (0 < _amount) {
            subPower(_pool, msg.sender, _amount, _upper1, _upper2);
        }

        setRewardDebt(_pool, poolInfo.rewardPerShare, msg.sender, _upper1, _upper2);
    }
    function computeReward(uint256 _pool) internal {
        PoolInfo storage poolInfo = poolInfos[_pool];
        if ((0 < poolInfo.totalPower) && (poolInfo.rewardProvide < poolInfo.rewardTotal)) {
            uint256 reward = (block.number -
poolInfo.lastRewardBlock).mul(poolInfo.rewardPerBlock);
```

```
                if (poolInfo.rewardProvide.add(reward) > poolInfo.rewardTotal) {
                    reward = poolInfo.rewardTotal.sub(poolInfo.rewardProvide);
                    poolInfo.endBlock = block.number;
                }

                rewardTotal = rewardTotal.add(reward);
                poolInfo.rewardProvide = poolInfo.rewardProvide.add(reward);
                poolInfo.rewardPerShare =
poolInfo.rewardPerShare.add(reward.mul(1e24).div(poolInfo.totalPower));
                poolInfo.lastRewardBlock = block.number;

                emit Mint(_pool, poolInfo.lp, reward);

                if (0 < poolInfo.endBlock) {
                    emit EndPool(_pool, poolInfo.lp);
                }
            }
        }
        function provideReward(uint256 _pool, uint256 _rewardPerShare, address _lp, address _user,
address _upper1, address _upper2) internal {
            uint256 inviteReward = 0;
            uint256 pledgeReward = 0;
            UserInfo storage userInfo = pledgeUserInfo[_pool][_user];
            if ((0 < userInfo.invitePower) || (0 < userInfo.pledgePower)) {
                inviteReward =
userInfo.invitePower.mul(_rewardPerShare).sub(userInfo.inviteRewardDebt).div(1e24);
                pledgeReward =
userInfo.pledgePower.mul(_rewardPerShare).sub(userInfo.pledgeRewardDebt).div(1e24);

                userInfo.pendingReward =
userInfo.pendingReward.add(inviteReward.add(pledgeReward));

                RewardInfo storage userRewardInfo = rewardInfos[_user];
```

```
            userRewardInfo.inviteReward = userRewardInfo.inviteReward.add(inviteReward);
            userRewardInfo.pledgeReward = userRewardInfo.pledgeReward.add(pledgeReward);
        }

        if (0 < userInfo.pendingReward) {
            you.mint(_user, userInfo.pendingReward);

            RewardInfo storage userRewardInfo = rewardInfos[_user];
            userRewardInfo.receiveReward = userRewardInfo.inviteReward;

            emit WithdrawReward(_pool, _lp, _user, userInfo.pendingReward);

            userInfo.pendingReward = 0;
        }

        if (address(0) != _upper1) {
            UserInfo storage upper1Info = pledgeUserInfo[_pool][_upper1];
            if ((0 < upper1Info.invitePower) || (0 < upper1Info.pledgePower)) {
                inviteReward =
upper1Info.invitePower.mul(_rewardPerShare).sub(upper1Info.inviteRewardDebt).div(1e24);
                pledgeReward =
upper1Info.pledgePower.mul(_rewardPerShare).sub(upper1Info.pledgeRewardDebt).div(1e24);

                upper1Info.pendingReward =
upper1Info.pendingReward.add(inviteReward.add(pledgeReward));

                RewardInfo storage upper1RewardInfo = rewardInfos[_upper1];
                upper1RewardInfo.inviteReward =
upper1RewardInfo.inviteReward.add(inviteReward);
                upper1RewardInfo.pledgeReward =
upper1RewardInfo.pledgeReward.add(pledgeReward);
            }
```

```
                if (address(0) != _upper2) {
                    UserInfo storage upper2Info = pledgeUserInfo[_pool][_upper2];
                    if ((0 < upper2Info.invitePower) || (0 < upper2Info.pledgePower)) {
                        inviteReward =
upper2Info.invitePower.mul(_rewardPerShare).sub(upper2Info.inviteRewardDebt).div(1e24);
                        pledgeReward =
upper2Info.pledgePower.mul(_rewardPerShare).sub(upper2Info.pledgeRewardDebt).div(1e24);

                        upper2Info.pendingReward =
upper2Info.pendingReward.add(inviteReward.add(pledgeReward));

                        RewardInfo storage upper2RewardInfo = rewardInfos[_upper2];
                        upper2RewardInfo.inviteReward =
upper2RewardInfo.inviteReward.add(inviteReward);
                        upper2RewardInfo.pledgeReward =
upper2RewardInfo.pledgeReward.add(pledgeReward);
                    }
                }
            }
        }
        function subPower(uint256 _pool, address _user, uint256 _amount, address _upper1, address
_upper2) internal {
            PoolInfo storage poolInfo = poolInfos[_pool];
            UserInfo storage userInfo = pledgeUserInfo[_pool][_user];
            poolInfo.amount = poolInfo.amount.sub(_amount);

            uint256 pledgePower = _amount;
            userInfo.amount = userInfo.amount.sub(_amount);
            userInfo.pledgePower = userInfo.pledgePower.sub(pledgePower);
            poolInfo.totalPower = poolInfo.totalPower.sub(pledgePower);

            uint256 upper1InvitePower = 0;
            uint256 upper2InvitePower = 0;
```

```
        if (address(0) != _upper1) {
            uint256 inviteSelfPower = pledgePower.mul(inviteSelfReward).div(100);
            userInfo.invitePower = userInfo.invitePower.sub(inviteSelfPower);
            poolInfo.totalPower = poolInfo.totalPower.sub(inviteSelfPower);

            UserInfo storage upper1Info = pledgeUserInfo[_pool][_upper1];
            if (0 < upper1Info.startBlock) {
                uint256 invite1Power = pledgePower.mul(invite1Reward).div(100);
                upper1Info.invitePower = upper1Info.invitePower.sub(invite1Power);
                upper1InvitePower = upper1Info.invitePower;
                poolInfo.totalPower = poolInfo.totalPower.sub(invite1Power);

                if (address(0) != _upper2) {
                    UserInfo storage upper2Info = pledgeUserInfo[_pool][_upper2];
                    if (0 < upper2Info.startBlock) {
                        uint256 invite2Power = pledgePower.mul(invite2Reward).div(100);
                        upper2Info.invitePower = upper2Info.invitePower.sub(invite2Power);
                        upper2InvitePower = upper2Info.invitePower;
                        poolInfo.totalPower = poolInfo.totalPower.sub(invite2Power);
                    }
                }
            }
        }

        emit UpdatePower(_pool, poolInfo.lp, poolInfo.totalPower, _user, userInfo.invitePower,
userInfo.pledgePower, _upper1, upper1InvitePower, _upper2, upper2InvitePower);
    }
    function setRewardDebt(uint256 _pool, uint256 _rewardPerShare, address _user, address _upper1,
address _upper2) internal {
        UserInfo storage userInfo = pledgeUserInfo[_pool][_user];
        userInfo.inviteRewardDebt = userInfo.invitePower.mul(_rewardPerShare);
        userInfo.pledgeRewardDebt = userInfo.pledgePower.mul(_rewardPerShare);
```

```
        if (address(0) != _upper1) {
            UserInfo storage upper1Info = pledgeUserInfo[_pool][_upper1];
            upper1Info.inviteRewardDebt = upper1Info.invitePower.mul(_rewardPerShare);
            upper1Info.pledgeRewardDebt = upper1Info.pledgePower.mul(_rewardPerShare);

            if (address(0) != _upper2) {
                UserInfo storage upper2Info = pledgeUserInfo[_pool][_upper2];
                upper2Info.inviteRewardDebt = upper2Info.invitePower.mul(_rewardPerShare);
                upper2Info.pledgeRewardDebt = upper2Info.pledgePower.mul(_rewardPerShare);
            }
        }
    }
```

安全建议：无。

3.6 算力占比计算逻辑【通过】

对 powerScale 算力占比计算逻辑设计进行审计，检查是否有对参数进行校验、函数调用者权限校验、以及相关逻辑设计是否合理。

审计结果：相关逻辑设计合理无误。

```
    function powerScale(uint256 _pool, address _user) override external view returns (uint256) {
        PoolInfo memory poolInfo = poolInfos[_pool];
        if (0 == poolInfo.totalPower) {
            return 0;
        }

        UserInfo memory userInfo = pledgeUserInfo[_pool][_user];
        return (userInfo.invitePower.add(userInfo.pledgePower).mul(100)).div(poolInfo.totalPower);
    }
```

安全建议：无。

3.7 待领取奖励逻辑设计【通过】

对 pendingReward 逻辑设计进行审计，检查是否有对参数进行校验、函数调用者权限校验、以及相关逻辑设计是否合理。

审计结果：相关逻辑设计合理无误。

```
    function pendingReward(uint256 _pool, address _user) override external view returns (uint256) {
        uint256 totalReward = 0;
        PoolInfo memory poolInfo = poolInfos[_pool];
        if (address(0) != poolInfo.lp && (poolInfo.startBlock <= block.number)) {
            uint256 rewardPerShare = 0;
            if (0 < poolInfo.totalPower) {
                uint256 reward = (block.number - poolInfo.lastRewardBlock).mul(poolInfo.rewardPerBlock);
                if (poolInfo.rewardProvide.add(reward) > poolInfo.rewardTotal) {
                    reward = poolInfo.rewardTotal.sub(poolInfo.rewardProvide);
                }
                rewardPerShare = reward.mul(1e24).div(poolInfo.totalPower);
            }
            rewardPerShare = rewardPerShare.add(poolInfo.rewardPerShare);

            UserInfo memory userInfo = pledgeUserInfo[_pool][_user];
            totalReward = userInfo.pendingReward;
            totalReward = totalReward.add(userInfo.invitePower.mul(rewardPerShare).sub(userInfo.inviteRewardDebt).div(1e24));
            totalReward = totalReward.add(userInfo.pledgePower.mul(rewardPerShare).sub(userInfo.pledgeRewardDebt).div(1e24));
        }
```

```
        return totalReward;
    }
```

安全建议：无。

3.8 下级收益贡献逻辑设计【通过】

对下级收益贡献逻辑设计进行审计，检查是否有对参数进行校验、函数调用者权限校验、以及相关逻辑设计是否合理。

审计结果：相关逻辑设计合理无误。

```
    function rewardContribute(address _user, address _lower) override external view returns (uint256)
{
        if ((address(0) == _user) || (address(0) == _lower)) {
            return 0;
        }

        uint256 inviteReward = 0;
        (address upper1, address upper2) = invite.inviteUpper2(_lower);
        if (_user == upper1) {
            inviteReward = rewardInfos[_lower].pledgeReward.mul(invite1Reward).div(100);
        }else if (_user == upper2) {
            inviteReward = rewardInfos[_lower].pledgeReward.mul(invite2Reward).div(100);
        }

        return inviteReward;
    }
```

安全建议：无。

3.9 个人收益加成逻辑【通过】

对个人收益加成逻辑设计进行审计，检查是否有对参数进行校验、函数调用者权限校验、以及相关逻辑设计是否合理。

审计结果：相关逻辑设计合理无误。

```
function selfReward(address _user) override external view returns (uint256) {
    address upper1 = invite.inviteUpper1(_user);
    if (address(0) == upper1) {
        return 0;
    }
    RewardInfo memory userRewardInfo = rewardInfos[_user];
    return userRewardInfo.pledgeReward.mul(inviteSelfReward).div(100);
}
```

安全建议：无。

3.10 新建矿池逻辑设计【通过】

对新建矿池逻辑设计进行审计，检查是否有对参数进行校验、函数调用者权限校验、以及相关逻辑设计是否合理。

审计结果：相关逻辑设计合理无误。

```
function addPool(string memory _name, address _lp, uint256 _startBlock, uint256 _rewardTotal) override external returns (bool) {
    require(operateOwner[msg.sender] && (address(0) != _lp) && (address(this) != _lp), ErrorCode.FORBIDDEN);
    _startBlock = _startBlock < block.number ? block.number : _startBlock;
    uint256 _pool = poolCount;
    poolCount = poolCount.add(1);

    PoolViewInfo storage poolViewInfo = poolViewInfos[_pool];
```

```
        poolViewInfo.lp = _lp;
        poolViewInfo.name = _name;
        poolViewInfo.multiple = 1;
        poolViewInfo.priority = _pool.mul(100);

        PoolInfo storage poolInfo = poolInfos[_pool];
        poolInfo.startBlock = _startBlock;
        poolInfo.rewardTotal = _rewardTotal;
        poolInfo.rewardProvide = 0;
        poolInfo.lp = _lp;
        poolInfo.amount = 0;
        poolInfo.lastRewardBlock = _startBlock.sub(1);
        poolInfo.rewardPerBlock = rewardPerBlock;
        poolInfo.totalPower = 0;
        poolInfo.endBlock = 0;
        poolInfo.rewardPerShare = 0;

        emit UpdatePool(true, _pool, poolInfo.lp, poolViewInfo.name, poolInfo.startBlock,
poolInfo.rewardTotal, poolInfo.rewardPerBlock, poolViewInfo.multiple, poolViewInfo.priority);

        return true;
    }
```

安全建议：无。

3.11 矿池信息修改逻辑【通过】

对矿池信息修改逻辑设计进行审计，检查是否有对参数进行校验、函数调用者权限校验、以及相关逻辑设计是否合理。

审计结果：相关逻辑设计合理无误。

```
    function setRewardPerBlock(uint256 _pool, uint256 _rewardPerBlock) override external {
        require(operateOwner[msg.sender], ErrorCode.FORBIDDEN);
```

```
        PoolInfo storage poolInfo = poolInfos[_pool];
        require((address(0) != poolInfo.lp) && (0 == poolInfo.endBlock), ErrorCode.POOL_NOT_EXIST_OR_END_OF_MINING);
        poolInfo.rewardPerBlock = _rewardPerBlock;

        PoolViewInfo memory poolViewInfo = poolViewInfos[_pool];

        emit UpdatePool(false, _pool, poolInfo.lp, poolViewInfo.name, poolInfo.startBlock, poolInfo.rewardTotal, poolInfo.rewardPerBlock, poolViewInfo.multiple, poolViewInfo.priority);
    }

    function setRewardTotal(uint256 _pool, uint256 _rewardTotal) override external {
        require(operateOwner[msg.sender], ErrorCode.FORBIDDEN);
        PoolInfo storage poolInfo = poolInfos[_pool];
        require((address(0) != poolInfo.lp) && (0 == poolInfo.endBlock), ErrorCode.POOL_NOT_EXIST_OR_END_OF_MINING);
        require(poolInfo.rewardProvide < _rewardTotal, ErrorCode.REWARDTOTAL_LESS_THAN_REWARDPROVIDE);
        poolInfo.rewardTotal = _rewardTotal;

        PoolViewInfo memory poolViewInfo = poolViewInfos[_pool];

        emit UpdatePool(false, _pool, poolInfo.lp, poolViewInfo.name, poolInfo.startBlock, poolInfo.rewardTotal, poolInfo.rewardPerBlock, poolViewInfo.multiple, poolViewInfo.priority);
    }

    function setName(uint256 _pool, string memory _name) override external {
        require(operateOwner[msg.sender], ErrorCode.FORBIDDEN);
        PoolViewInfo storage poolViewInfo = poolViewInfos[_pool];
        require(address(0) != poolViewInfo.lp, ErrorCode.POOL_NOT_EXIST_OR_END_OF_MINING);
        poolViewInfo.name = _name;
```

```
        PoolInfo memory poolInfo = poolInfos[_pool];

        emit UpdatePool(false, _pool, poolInfo.lp, poolViewInfo.name, poolInfo.startBlock,
poolInfo.rewardTotal, poolInfo.rewardPerBlock, poolViewInfo.multiple, poolViewInfo.priority);
    }

    function setMultiple(uint256 _pool, uint256 _multiple) override external {
        require(operateOwner[msg.sender], ErrorCode.FORBIDDEN);
        PoolViewInfo storage poolViewInfo = poolViewInfos[_pool];
        require(address(0) != poolViewInfo.lp,
ErrorCode.POOL_NOT_EXIST_OR_END_OF_MINING);
        poolViewInfo.multiple = _multiple;

        PoolInfo memory poolInfo = poolInfos[_pool];

        emit UpdatePool(false, _pool, poolInfo.lp, poolViewInfo.name, poolInfo.startBlock,
poolInfo.rewardTotal, poolInfo.rewardPerBlock, poolViewInfo.multiple, poolViewInfo.priority);
    }

    function setPriority(uint256 _pool, uint256 _priority) override external {
        require(operateOwner[msg.sender], ErrorCode.FORBIDDEN);
        PoolViewInfo storage poolViewInfo = poolViewInfos[_pool];
        require(address(0) != poolViewInfo.lp,
ErrorCode.POOL_NOT_EXIST_OR_END_OF_MINING);
        poolViewInfo.priority = _priority;

        PoolInfo memory poolInfo = poolInfos[_pool];

        emit UpdatePool(false, _pool, poolInfo.lp, poolViewInfo.name, poolInfo.startBlock,
poolInfo.rewardTotal, poolInfo.rewardPerBlock, poolViewInfo.multiple, poolViewInfo.priority);
    }
```

安全建议：无。

4. 代码基本漏洞检测

4.1. 编译器版本安全【通过】

检查合约代码实现中是否使用了安全的编译器版本

检测结果：经检测，智能合约代码中制定了编译器版本 0.5.15 以上，不存在该安全问题。

安全建议：无。

4.2. 冗余代码【通过】

检查合约代码实现中是否包含冗余代码

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.3. 安全算数库的使用【通过】

检查合约代码实现中是否使用了 SafeMath 安全算数库

检测结果：经检测，智能合约代码中已使用 SafeMath 安全算数库，不存在该安全问题。

安全建议：无。

4.4. 不推荐的编码方式【通过】

检查合约代码实现中是否有官方不推荐或弃用的编码方式

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.5. require/assert 的合理使用【通过】

检查合约代码实现中 require 和 assert 语句使用的合理性

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.6. fallback 函数安全【通过】

检查合约代码实现中是否正确使用 fallback 函数

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.7. tx.origin 身份验证【通过】

tx.origin 是 Solidity 的一个全局变量，它遍历整个调用栈并返回最初发送调用（或事务）的帐户的地址。在智能合约中使用此变量进行身份验证会使合约容易受到类似网络钓鱼的攻击。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.8. owner 权限控制【通过】

检查合约代码实现中的 owner 是否具有过高的权限。例如，任意修改其他账户余额等。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.9. gas 消耗检测【通过】

检查 gas 的消耗是否超过区块最大限制

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.10. call 注入攻击【通过】

call 函数调用时，应该做严格的权限控制，或直接写死 call 调用的函数。

检测结果：经检测，智能合约未使用 call 函数，不存在此漏洞。

安全建议：无。

4.11. 低级函数安全【通过】

检查合约代码实现中低级函数（call/delegatecall）的使用是否存在安全漏洞

call 函数的执行上下文是在被调用的合约中；而 delegatecall 函数的执行上下文是在当前调用该函数的合约中

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.12. 增发代币漏洞【通过】

检查在初始化代币总量后，代币合约中是否存在可能使代币总量增加的函数。

检测结果：经检测，智能合约代码中不能存在增发代币的功能，通过。

安全建议：无。

4.13. 访问控制缺陷检测【通过】

合约中不同函数应设置合理的权限

检查合约中各函数是否正确使用了 public、private 等关键词进行可见性修饰，检查合约是否正确定义并使用了 modifier 对关键函数进行访问限制，避免越权导致的问题。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.14. 数值溢出检测【通过】

智能合约中的算数问题是指整数溢出和整数下溢。

Solidity 最多能处理 256 位的数字（2^256-1），最大数字增加 1 会溢出得到 0。同样，当数字为无符号类型时，0 减去 1 会下溢得到最大数字值。

整数溢出和下溢不是一种新类型的漏洞，但它们在智能合约中尤其危险。溢出情况会导致不正确的结果，特别是如果可能性未被预期，可能会影响程序的可靠性和安全性。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.15. 算术精度误差【通过】

Solidity 作为一门编程语言具备和普通编程语言相似的数据结构设计，比如：变量、常量、函数、数组、函数、结构体等等，Solidity 和普通编程语言也有一个较大的区别——Solidity 没有浮点型，且 Solidity 所有的数值运算结果都只会是整数，不会出现小数的情况，同时也不允许定义小数类型数据。合约中的数值运算必不可少，而数值运算的设计有可能造成相对误差，例如同级运算：5/2*10=20，而 5*10/2=25，从而产生误差，在数据更大时产生的误差也会更大，更明显。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.16. 错误使用随机数【通过】

智能合约中可能需要使用随机数，虽然 Solidity 提供的函数和变量可以访问明显难以预测的值，如 block.number 和 block.timestamp，但是它们通常或者比看起来更公开，或者受到矿工的影响，即这些随机数在一定程度上是可预测的，所以恶意用户通常可以复制它并依靠其不可预知性来攻击该功能。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.17. 不安全的接口使用【通过】

检查合约代码实现中是否使用了不安全的接口

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.18. 变量覆盖【通过】

检查合约代码实现中是否存在变量覆盖导致的安全问题

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.19. 未初始化的储存指针【通过】

在 solidity 中允许一个特殊的数据结构为 struct 结构体，而函数内的局部变量默认使用 storage 或 memory 储存。

而存在 storage(存储器)和 memory(内存)是两个不同的概念，solidity 允许指针指向一个未初始化的引用，而未初始化的局部 stroage 会导致变量指向其他储存变量，导致变量覆盖，甚至其他更严重的后果，在开发中应该避免在函数中初始化 struct 变量。

检测结果：经检测，智能合约代码不使用结构体，不存在该问题。

安全建议：无。

4.20. 返回值调用验证【通过】

此问题多出现在和转币相关的智能合约中，故又称作静默失败发送或未经检查发送。

在 Solidity 中存在 transfer()、send()、call.value()等转币方法，都可以用于向某一地址发送 Ether，其区别在于： transfer 发送失败时会 throw，并且进行状态回滚；只会传递 2300gas 供调用，防止重入攻击；send 发送失败时会返回 false；只会传递 2300gas 供调用，防止重入攻击；call.value 发送失败时会返回 false；

传递所有可用 gas 进行调用（可通过传入 gas_value 参数进行限制），不能有效防止重入攻击。

如果在代码中没有检查以上 send 和 call.value 转币函数的返回值，合约会继续执行后面的代码，可能由于 Ether 发送失败而导致意外的结果。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.21. 交易顺序依赖【通过】

由于矿工总是通过代表外部拥有地址（EOA）的代码获取 gas 费用，因此用户可以指定更高的费用以便更快地开展交易。由于以太坊区块链是公开的，每个人都可以看到其他人未决交易的内容。这意味着，如果某个用户提交了一个有价值的解决方案，恶意用户可以窃取该解决方案并以较高的费用复制其交易，以抢占原始解决方案。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.22. 时间戳依赖攻击【通过】

数据块的时间戳通常来说都是使用矿工的本地时间，而这个时间大约能有 900 秒的范围波动，当其他节点接受一个新区块时，只需要验证时间戳是否晚于之前的区块并且与本地时间误差在 900 秒以内。一个矿工可以通过设置区块的时间戳来尽可能满足有利于他的条件来从中获利。

检查合约代码实现中是否存在有依赖于时间戳的关键功能

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.23. 拒绝服务攻击【通过】

在以太坊的世界中，拒绝服务是致命的，遭受该类型攻击的智能合约可能永远无法恢复正常工作状态。导致智能合约拒绝服务的原因可能有很多种，包括在作为交易接收方时的恶意行为，人为增加计算功能所需 gas 导致 gas 耗尽，滥用访问控制访问智能合约的 private 组件，利用混淆和疏忽等等。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.24. 假充值漏洞【通过】

在代币合约的 transfer 函数对转账发起人(msg.sender)的余额检查用的是 if 判断方式，当 balances[msg.sender] < value 时进入 else 逻辑部分并 return false，最终没有抛出异常，我们认为仅 if/else 这种温和的判断方式在 transfer 这类敏感函数场景中是一种不严谨的编码方式。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.25. 重入攻击检测【通过】

Solidity 中的 call.value()函数在被用来发送 Ether 的时候会消耗它接收到的所有 gas，当调用 call.value()函数发送 Ether 的操作发生在实际减少发送者账户的

余额之前时，就会存在重入攻击的风险。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.26. 重放攻击检测【通过】

合约中如果涉及委托管理的需求，应注意验证的不可复用性，避免重放攻击

在资产管理体系中，常有委托管理的情况，委托人将资产给受托人管理，委托人支付一定的费用给受托人。这个业务场景在智能合约中也比较普遍。。

检测结果：经检测，智能合约未使用 call 函数，不存在此漏洞。

安全建议：无。

4.27. 重排攻击检测【通过】

重排攻击是指矿工或其他方试图通过将自己的信息插入列表(list)或映射(mapping)中来与智能合约参与者进行“竞争”，从而使攻击者有机会将自己的信息存储到合约中。

检测结果:经检测，智能合约代码中不存在相关漏洞。

安全建议:无。

5. 附录 A：合约代码

本次测试代码来源：

Youswap.sol

```
/**
 *Submitted for verification at Etherscan.io on 2021-03-26
*/

/**
 *Submitted for verification at Etherscan.io on 2021-03-26
*/

// File: @openzeppelin/contracts/utils/Address.sol

// SPDX-License-Identifier: MIT

pragma solidity >=0.6.2 <0.8.0;

/**
 * @dev Collection of functions related to the address type
 */
library Address {
    /**
     * @dev Returns true if `account` is a contract.
     *
     * [IMPORTANT]
     * ====
     * It is unsafe to assume that an address for which this function returns
     * false is an externally-owned account (EOA) and not a contract.
     *
     * Among others, `isContract` will return false for the following
     * types of addresses:
```

```
     *
     *  - an externally-owned account
     *  - a contract in construction
     *  - an address where a contract will be created
     *  - an address where a contract lived, but was destroyed
     * ====
     */
    function isContract(address account) internal view returns (bool) {
        // This method relies on extcodesize, which returns 0 for contracts in
        // construction, since the code is only stored at the end of the
        // constructor execution.

        uint256 size;
        // solhint-disable-next-line no-inline-assembly
        assembly { size := extcodesize(account) }
        return size > 0;
    }

    /**
     * @dev Replacement for Solidity's `transfer`: sends `amount` wei to
     * `recipient`, forwarding all available gas and reverting on errors.
     *
     * https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas cost
     * of certain opcodes, possibly making contracts go over the 2300 gas limit
     * imposed by `transfer`, making them unable to receive funds via
     * `transfer`. {sendValue} removes this limitation.
     *
     * https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-transfer-now/[Learn more].
     *
     * IMPORTANT: because control is transferred to `recipient`, care must be
     * taken to not create reentrancy vulnerabilities. Consider using
     * {ReentrancyGuard} or the
     *  https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-checks-effects-
```

```
interactions-pattern[checks-effects-interactions pattern].
     */
    function sendValue(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, "Address: insufficient balance");

        // solhint-disable-next-line avoid-low-level-calls, avoid-call-value
        (bool success, ) = recipient.call{ value: amount }("");
        require(success, "Address: unable to send value, recipient may have reverted");
    }

    /**
     * @dev Performs a Solidity function call using a low level `call`. A
     * plain`call` is an unsafe replacement for a function call: use this
     * function instead.
     *
     * If `target` reverts with a revert reason, it is bubbled up by this
     * function (like regular Solidity function calls).
     *
     * Returns the raw returned data. To convert to the expected return value,
     *                 use                 https://solidity.readthedocs.io/en/latest/units-and-global-
variables.html?highlight=abi.decode#abi-encoding-and-decoding-functions[`abi.decode`].
     *
     * Requirements:
     *
     * - `target` must be a contract.
     * - calling `target` with `data` must not revert.
     *
     * _Available since v3.1._
     */
    function functionCall(address target, bytes memory data) internal returns (bytes memory) {
      return functionCall(target, data, "Address: low-level call failed");
    }
```

```
    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`], but with
     * `errorMessage` as a fallback revert reason when `target` reverts.
     *
     * _Available since v3.1._
     */
    function functionCall(address target, bytes memory data, string memory errorMessage) internal returns (bytes memory) {
        return functionCallWithValue(target, data, 0, errorMessage);
    }

    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
     * but also transferring `value` wei to `target`.
     *
     * Requirements:
     *
     * - the calling contract must have an ETH balance of at least `value`.
     * - the called Solidity function must be `payable`.
     *
     * _Available since v3.1._
     */
    function functionCallWithValue(address target, bytes memory data, uint256 value) internal returns (bytes memory) {
        return functionCallWithValue(target, data, value, "Address: low-level call with value failed");
    }

    /**
     * @dev Same as {xref-Address-functionCallWithValue-address-bytes-uint256-}[`functionCallWithValue`], but
     * with `errorMessage` as a fallback revert reason when `target` reverts.
     *
     * _Available since v3.1._
```

```
     */
    function functionCallWithValue(address target, bytes memory data, uint256 value, string memory errorMessage) internal returns (bytes memory) {
        require(address(this).balance >= value, "Address: insufficient balance for call");
        require(isContract(target), "Address: call to non-contract");

        // solhint-disable-next-line avoid-low-level-calls
        (bool success, bytes memory returndata) = target.call{ value: value }(data);
        return _verifyCallResult(success, returndata, errorMessage);
    }

    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
     * but performing a static call.
     *
     * _Available since v3.3._
     */
    function functionStaticCall(address target, bytes memory data) internal view returns (bytes memory) {
        return functionStaticCall(target, data, "Address: low-level static call failed");
    }

    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-string-}[`functionCall`],
     * but performing a static call.
     *
     * _Available since v3.3._
     */
    function functionStaticCall(address target, bytes memory data, string memory errorMessage) internal view returns (bytes memory) {
        require(isContract(target), "Address: static call to non-contract");

        // solhint-disable-next-line avoid-low-level-calls
```

```
        (bool success, bytes memory returndata) = target.staticcall(data);
        return _verifyCallResult(success, returndata, errorMessage);
    }

    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
     * but performing a delegate call.
     *
     * _Available since v3.4._
     */
    function functionDelegateCall(address target, bytes memory data) internal returns (bytes memory)
{
        return functionDelegateCall(target, data, "Address: low-level delegate call failed");
    }

    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-string-}[`functionCall`],
     * but performing a delegate call.
     *
     * _Available since v3.4._
     */
    function functionDelegateCall(address target, bytes memory data, string memory errorMessage)
internal returns (bytes memory) {
        require(isContract(target), "Address: delegate call to non-contract");

        // solhint-disable-next-line avoid-low-level-calls
        (bool success, bytes memory returndata) = target.delegatecall(data);
        return _verifyCallResult(success, returndata, errorMessage);
    }

    function _verifyCallResult(bool success, bytes memory returndata, string memory errorMessage)
private pure returns(bytes memory) {
        if (success) {
```

```
                return returndata;
            } else {
                // Look for revert reason and bubble it up if present
                if (returndata.length > 0) {
                    // The easiest way to bubble the revert reason is using memory via assembly

                    // solhint-disable-next-line no-inline-assembly
                    assembly {
                        let returndata_size := mload(returndata)
                        revert(add(32, returndata), returndata_size)
                    }
                } else {
                    revert(errorMessage);
                }
            }
        }
}

// File: @openzeppelin/contracts/math/SafeMath.sol



pragma solidity >=0.6.0 <0.8.0;

/**
 * @dev Wrappers over Solidity's arithmetic operations with added overflow
 * checks.
 *
 * Arithmetic operations in Solidity wrap on overflow. This can easily result
 * in bugs, because programmers usually assume that an overflow raises an
 * error, which is the standard behavior in high level programming languages.
 * `SafeMath` restores this intuition by reverting the transaction when an
 * operation overflows.
```

```
 *
 * Using this library instead of the unchecked operations eliminates an entire
 * class of bugs, so it's recommended to use it always.
 */
library SafeMath {
    /**
     * @dev Returns the addition of two unsigned integers, with an overflow flag.
     *
     * _Available since v3.4._
     */
    function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint256) {
        uint256 c = a + b;
        if (c < a) return (false, 0);
        return (true, c);
    }

    /**
     * @dev Returns the substraction of two unsigned integers, with an overflow flag.
     *
     * _Available since v3.4._
     */
    function trySub(uint256 a, uint256 b) internal pure returns (bool, uint256) {
        if (b > a) return (false, 0);
        return (true, a - b);
    }

    /**
     * @dev Returns the multiplication of two unsigned integers, with an overflow flag.
     *
     * _Available since v3.4._
     */
    function tryMul(uint256 a, uint256 b) internal pure returns (bool, uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
```

```
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
        if (a == 0) return (true, 0);
        uint256 c = a * b;
        if (c / a != b) return (false, 0);
        return (true, c);
    }

    /**
     * @dev Returns the division of two unsigned integers, with a division by zero flag.
     *
     * _Available since v3.4._
     */
    function tryDiv(uint256 a, uint256 b) internal pure returns (bool, uint256) {
        if (b == 0) return (false, 0);
        return (true, a / b);
    }

    /**
     * @dev Returns the remainder of dividing two unsigned integers, with a division by zero flag.
     *
     * _Available since v3.4._
     */
    function tryMod(uint256 a, uint256 b) internal pure returns (bool, uint256) {
        if (b == 0) return (false, 0);
        return (true, a % b);
    }

    /**
     * @dev Returns the addition of two unsigned integers, reverting on
     * overflow.
     *
     * Counterpart to Solidity's `+` operator.
```

```
     *
     * Requirements:
     *
     * - Addition cannot overflow.
     */
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");
        return c;
    }

    /**
     * @dev Returns the subtraction of two unsigned integers, reverting on
     * overflow (when the result is negative).
     *
     * Counterpart to Solidity's `-` operator.
     *
     * Requirements:
     *
     * - Subtraction cannot overflow.
     */
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b <= a, "SafeMath: subtraction overflow");
        return a - b;
    }

    /**
     * @dev Returns the multiplication of two unsigned integers, reverting on
     * overflow.
     *
     * Counterpart to Solidity's `*` operator.
     *
     * Requirements:
```

```
     *
     * - Multiplication cannot overflow.
     */
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) return 0;
        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplication overflow");
        return c;
    }

    /**
     * @dev Returns the integer division of two unsigned integers, reverting on
     * division by zero. The result is rounded towards zero.
     *
     * Counterpart to Solidity's `/` operator. Note: this function uses a
     * `revert` opcode (which leaves remaining gas untouched) while Solidity
     * uses an invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0, "SafeMath: division by zero");
        return a / b;
    }

    /**
     * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),
     * reverting when dividing by zero.
     *
     * Counterpart to Solidity's `%` operator. This function uses a `revert`
     * opcode (which leaves remaining gas untouched) while Solidity uses an
```

```
     * invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0, "SafeMath: modulo by zero");
        return a % b;
    }

    /**
     * @dev Returns the subtraction of two unsigned integers, reverting with custom message on
     * overflow (when the result is negative).
     *
     * CAUTION: This function is deprecated because it requires allocating memory for the error
     * message unnecessarily. For custom revert reasons use {trySub}.
     *
     * Counterpart to Solidity's `-` operator.
     *
     * Requirements:
     *
     * - Subtraction cannot overflow.
     */
    function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        require(b <= a, errorMessage);
        return a - b;
    }

    /**
     * @dev Returns the integer division of two unsigned integers, reverting with custom message on
     * division by zero. The result is rounded towards zero.
     *
```

```
     * CAUTION: This function is deprecated because it requires allocating memory for the error
     * message unnecessarily. For custom revert reasons use {tryDiv}.
     *
     * Counterpart to Solidity's `/` operator. Note: this function uses a
     * `revert` opcode (which leaves remaining gas untouched) while Solidity
     * uses an invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        require(b > 0, errorMessage);
        return a / b;
    }

    /**
     * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),
     * reverting with custom message when dividing by zero.
     *
     * CAUTION: This function is deprecated because it requires allocating memory for the error
     * message unnecessarily. For custom revert reasons use {tryMod}.
     *
     * Counterpart to Solidity's `%` operator. This function uses a `revert`
     * opcode (which leaves remaining gas untouched) while Solidity uses an
     * invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        require(b > 0, errorMessage);
```

```
            return a % b;
        }
}

// File: @openzeppelin/contracts/token/ERC20/IERC20.sol



pragma solidity >=0.6.0 <0.8.0;

/**
 * @dev Interface of the ERC20 standard as defined in the EIP.
 */
interface IERC20 {
    /**
     * @dev Returns the amount of tokens in existence.
     */
    function totalSupply() external view returns (uint256);

    /**
     * @dev Returns the amount of tokens owned by `account`.
     */
    function balanceOf(address account) external view returns (uint256);

    /**
     * @dev Moves `amount` tokens from the caller's account to `recipient`.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transfer(address recipient, uint256 amount) external returns (bool);
```

```
/**
 * @dev Returns the remaining number of tokens that `spender` will be
 * allowed to spend on behalf of `owner` through {transferFrom}. This is
 * zero by default.
 *
 * This value changes when {approve} or {transferFrom} are called.
 */
function allowance(address owner, address spender) external view returns (uint256);

/**
 * @dev Sets `amount` as the allowance of `spender` over the caller's tokens.
 *
 * Returns a boolean value indicating whether the operation succeeded.
 *
 * IMPORTANT: Beware that changing an allowance with this method brings the risk
 * that someone may use both the old and the new allowance by unfortunate
 * transaction ordering. One possible solution to mitigate this race
 * condition is to first reduce the spender's allowance to 0 and set the
 * desired value afterwards:
 * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
 *
 * Emits an {Approval} event.
 */
function approve(address spender, uint256 amount) external returns (bool);

/**
 * @dev Moves `amount` tokens from `sender` to `recipient` using the
 * allowance mechanism. `amount` is then deducted from the caller's
 * allowance.
 *
 * Returns a boolean value indicating whether the operation succeeded.
 *
 * Emits a {Transfer} event.
```

```
     */
    function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);

    /**
     * @dev Emitted when `value` tokens are moved from one account (`from`) to
     * another (`to`).
     *
     * Note that `value` may be zero.
     */
    event Transfer(address indexed from, address indexed to, uint256 value);

    /**
     * @dev Emitted when the allowance of a `spender` for an `owner` is set by
     * a call to {approve}. `value` is the new allowance.
     */
    event Approval(address indexed owner, address indexed spender, uint256 value);
}

// File: @openzeppelin/contracts/token/ERC20/SafeERC20.sol


pragma solidity >=0.6.0 <0.8.0;




/**
 * @title SafeERC20
 * @dev Wrappers around ERC20 operations that throw on failure (when the token
 * contract returns false). Tokens that return no value (and instead revert or
 * throw on failure) are also supported, non-reverting calls are assumed to be
 * successful.
```

```
 * To use this library you can add a `using SafeERC20 for IERC20;` statement to your contract,
 * which allows you to call the safe operations as `token.safeTransfer(...)`, etc.
 */
library SafeERC20 {
    using SafeMath for uint256;
    using Address for address;

    function safeTransfer(IERC20 token, address to, uint256 value) internal {
        _callOptionalReturn(token, abi.encodeWithSelector(token.transfer.selector, to, value));
    }

    function safeTransferFrom(IERC20 token, address from, address to, uint256 value) internal {
        _callOptionalReturn(token, abi.encodeWithSelector(token.transferFrom.selector, from, to,
value));
    }

    /**
     * @dev Deprecated. This function has issues similar to the ones found in
     * {IERC20-approve}, and its usage is discouraged.
     *
     * Whenever possible, use {safeIncreaseAllowance} and
     * {safeDecreaseAllowance} instead.
     */
    function safeApprove(IERC20 token, address spender, uint256 value) internal {
        // safeApprove should only be called when setting an initial allowance,
        // or when resetting it to zero. To increase and decrease it, use
        // 'safeIncreaseAllowance' and 'safeDecreaseAllowance'
        // solhint-disable-next-line max-line-length
        require((value == 0) || (token.allowance(address(this), spender) == 0),
            "SafeERC20: approve from non-zero to non-zero allowance"
        );
        _callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, value));
    }
```

```
    function safeIncreaseAllowance(IERC20 token, address spender, uint256 value) internal {
        uint256 newAllowance = token.allowance(address(this), spender).add(value);
        _callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, newAllowance));
    }

    function safeDecreaseAllowance(IERC20 token, address spender, uint256 value) internal {
        uint256 newAllowance = token.allowance(address(this), spender).sub(value, "SafeERC20: decreased allowance below zero");
        _callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, newAllowance));
    }

    /**
     * @dev Imitates a Solidity high-level call (i.e. a regular function call to a contract), relaxing the requirement
     * on the return value: the return value is optional (but if data is returned, it must not be false).
     * @param token The token targeted by the call.
     * @param data The call data (encoded using abi.encode or one of its variants).
     */
    function _callOptionalReturn(IERC20 token, bytes memory data) private {
        // We need to perform a low level call here, to bypass Solidity's return data size checking mechanism, since
        // we're implementing it ourselves. We use {Address.functionCall} to perform this call, which verifies that
        // the target address contains contract code and also asserts for success in the low-level call.

        bytes memory returndata = address(token).functionCall(data, "SafeERC20: low-level call failed");
        if (returndata.length > 0) { // Return data is optional
            // solhint-disable-next-line max-line-length
            require(abi.decode(returndata, (bool)), "SafeERC20: ERC20 operation did not succeed");
```

```
            }
        }
}

// File: localhost/contract/library/ErrorCode.sol



pragma solidity 0.7.4;

library ErrorCode {

    string constant FORBIDDEN = 'YouSwap:FORBIDDEN';
    string constant IDENTICAL_ADDRESSES = 'YouSwap:IDENTICAL_ADDRESSES';
    string constant ZERO_ADDRESS = 'YouSwap:ZERO_ADDRESS';
    string constant INVALID_ADDRESSES = 'YouSwap:INVALID_ADDRESSES';
    string constant BALANCE_INSUFFICIENT = 'YouSwap:BALANCE_INSUFFICIENT';
    string constant REWARDTOTAL_LESS_THAN_REWARDPROVIDE =
'YouSwap:REWARDTOTAL_LESS_THAN_REWARDPROVIDE';
    string constant PARAMETER_TOO_LONG = 'YouSwap:PARAMETER_TOO_LONG';
    string constant REGISTERED = 'YouSwap:REGISTERED';
    string constant MINING_NOT_STARTED = 'YouSwap:MINING_NOT_STARTED';
    string constant END_OF_MINING = 'YouSwap:END_OF_MINING';
    string constant POOL_NOT_EXIST_OR_END_OF_MINING =
'YouSwap:POOL_NOT_EXIST_OR_END_OF_MINING';

}
// File: localhost/contract/interface/IYouswapInviteV1.sol



pragma solidity 0.7.4;
```

```
interface IYouswapInviteV1 {

    struct UserInfo {
        address upper;//上级
        address[] lowers;//下级
        uint256 startBlock;//邀请块高
    }

    event InviteV1(address indexed owner, address indexed upper, uint256 indexed height);//被邀请人的地址，邀请人的地址，邀请块高

    function inviteCount() external view returns (uint256);//邀请人数

    function inviteUpper1(address) external view returns (address);//上级邀请

    function inviteUpper2(address) external view returns (address, address);//上级邀请

    function inviteLower1(address) external view returns (address[] memory);//下级邀请

    function inviteLower2(address) external view returns (address[] memory, address[] memory);//下级邀请

    function inviteLower2Count(address) external view returns (uint256, uint256);//下级邀请

    function register() external returns (bool);//注册邀请关系

    function acceptInvitation(address) external returns (bool);//注册邀请关系

    function inviteBatch(address[] memory) external returns (uint, uint);//注册邀请关系：输入数量，成功数量

}
// File: localhost/contract/implement/YouswapInviteV1.sol
```

```
pragma solidity 0.7.4;



contract YouswapInviteV1 is IYouswapInviteV1 {

    address public constant ZERO = address(0);
    uint256 public startBlock;
    address[] public inviteUserInfoV1;
    mapping(address => UserInfo) public inviteUserInfoV2;

    constructor () {
        startBlock = block.number;
    }

    function inviteCount() override external view returns (uint256) {
        return inviteUserInfoV1.length;
    }

    function inviteUpper1(address _owner) override external view returns (address) {
        return inviteUserInfoV2[_owner].upper;
    }

    function inviteUpper2(address _owner) override external view returns (address, address) {
        address upper1 = inviteUserInfoV2[_owner].upper;
        address upper2 = address(0);
        if (address(0) != upper1) {
            upper2 = inviteUserInfoV2[upper1].upper;
        }
```

```
        return (upper1, upper2);
    }

    function inviteLower1(address _owner) override external view returns (address[] memory) {
        return inviteUserInfoV2[_owner].lowers;
    }

    function inviteLower2(address _owner) override external view returns (address[] memory, address[] memory) {
        address[] memory lowers1 = inviteUserInfoV2[_owner].lowers;
        uint256 count = 0;
        uint256 lowers1Len = lowers1.length;
        for (uint256 i = 0; i < lowers1Len; i++) {
            count += inviteUserInfoV2[lowers1[i]].lowers.length;
        }
        address[] memory lowers;
        address[] memory lowers2 = new address[](count);
        count = 0;
        for (uint256 i = 0; i < lowers1Len; i++) {
            lowers = inviteUserInfoV2[lowers1[i]].lowers;
            for (uint256 j = 0; j < lowers.length; j++) {
                lowers2[count] = lowers[j];
                count++;
            }
        }

        return (lowers1, lowers2);
    }

    function inviteLower2Count(address _owner) override external view returns (uint256, uint256) {
        address[] memory lowers1 = inviteUserInfoV2[_owner].lowers;
        uint256 lowers2Len = 0;
        uint256 len = lowers1.length;
```

```
        for (uint256 i = 0; i < len; i++) {
            lowers2Len += inviteUserInfoV2[lowers1[i]].lowers.length;
        }

        return (lowers1.length, lowers2Len);
    }

    function register() override external returns (bool) {
        UserInfo storage user = inviteUserInfoV2[tx.origin];
        require(0 == user.startBlock, ErrorCode.REGISTERED);
        user.upper = ZERO;
        user.startBlock = block.number;
        inviteUserInfoV1.push(tx.origin);

        emit InviteV1(tx.origin, user.upper, user.startBlock);

        return true;
    }

    function acceptInvitation(address _inviter) override external returns (bool) {
        require(msg.sender != _inviter, ErrorCode.FORBIDDEN);
        UserInfo storage user = inviteUserInfoV2[msg.sender];
        require(0 == user.startBlock, ErrorCode.REGISTERED);
        UserInfo storage upper = inviteUserInfoV2[_inviter];
        if (0 == upper.startBlock) {
            upper.upper = ZERO;
            upper.startBlock = block.number;
            inviteUserInfoV1.push(_inviter);

            emit InviteV1(_inviter, upper.upper, upper.startBlock);
        }
        user.upper = _inviter;
        upper.lowers.push(msg.sender);
```

```
        user.startBlock = block.number;
        inviteUserInfoV1.push(msg.sender);

        emit InviteV1(msg.sender, user.upper, user.startBlock);

        return true;
    }

    function inviteBatch(address[] memory _invitees) override external returns (uint, uint) {
        uint len = _invitees.length;
        require(len <= 100, ErrorCode.PARAMETER_TOO_LONG);
        UserInfo storage user = inviteUserInfoV2[msg.sender];
        if (0 == user.startBlock) {
            user.upper = ZERO;
            user.startBlock = block.number;
            inviteUserInfoV1.push(msg.sender);

            emit InviteV1(msg.sender, user.upper, user.startBlock);
        }
        uint count = 0;
        for (uint i = 0; i < len; i++) {
            if ((address(0) != _invitees[i]) && (msg.sender != _invitees[i])) {
                UserInfo storage lower = inviteUserInfoV2[_invitees[i]];
                if (0 == lower.startBlock) {
                    lower.upper = msg.sender;
                    lower.startBlock = block.number;
                    user.lowers.push(_invitees[i]);
                    inviteUserInfoV1.push(_invitees[i]);
                    count++;

                    emit InviteV1(_invitees[i], msg.sender, lower.startBlock);
                }
            }
```

```
            }

            return (len, count);
        }

}
// File: localhost/contract/interface/ITokenYou.sol



pragma solidity 0.7.4;

interface ITokenYou {

        function mint(address recipient, uint256 amount) external;

        function decimals() external view returns (uint8);

}

// File: localhost/contract/interface/IYouswapFactoryV1.sol


pragma solidity 0.7.4;



/**
挖矿
 */
interface IYouswapFactoryV1 {
```

```
/**
用户挖矿信息
 */
struct RewardInfo {
    uint256 receiveReward;//总领取奖励
    uint256 inviteReward;//总邀请奖励
    uint256 pledgeReward;//总质押奖励
}

/**
质押用户信息
 */
struct UserInfo {
    uint256 startBlock;//质押开始块高
    uint256 amount;//质押数量
    uint256 invitePower;//邀请算力
    uint256 pledgePower;//质押算力
    uint256 pendingReward;//待领取奖励
    uint256 inviteRewardDebt;//邀请负债
    uint256 pledgeRewardDebt;//质押负债
}

/**
矿池信息（可视化）
 */
struct PoolViewInfo {
    address lp;//LP 地址
    string name;//名称
    uint256 multiple;//奖励倍数
    uint256 priority;//排序
}

/**
```

```
矿池信息
 */
struct PoolInfo {
        uint256 startBlock;//挖矿开始块高
        uint256 rewardTotal;//矿池总奖励
        uint256 rewardProvide;//矿池已发放奖励
        address lp;//lp 合约地址
        uint256 amount;//质押数量
        uint256 lastRewardBlock;//最后发放奖励块高
        uint256 rewardPerBlock;//单个区块奖励
        uint256 totalPower;//总算力
        uint256 endBlock;//挖矿结束块高
        uint256 rewardPerShare;//单位算力奖励
}

////////////////////////////////////////////////////////////////////////////////

/**
自邀请
self：Sender 地址
 */
event InviteRegister(address indexed self);

/**
更新矿池信息

action：true(新建矿池)，false(更新矿池)
pool：矿池序号
lp：lp 合约地址
name：矿池名称
startBlock：矿池开始挖矿块高
rewardTotal：矿池总奖励
rewardPerBlock：区块奖励
```

multiple：矿池奖励倍数

priority：矿池排序

*/

event UpdatePool(bool action, uint256 pool, address indexed lp, string name, uint256 startBlock, uint256 rewardTotal, uint256 rewardPerBlock, uint256 multiple, uint256 priority);

/**

矿池挖矿结束

pool：矿池序号

lp：lp 合约地址

*/

event EndPool(uint256 pool, address indexed lp);

/**

质押

pool：矿池序号

lp：lp 合约地址

from：质押转出地址

amount：质押数量

*/

event Stake(uint256 pool, address indexed lp, address indexed from, uint256 amount);

/**

pool：矿池序号

lp：lp 合约地址

totalPower：矿池总算力

owner：用户地址

ownerInvitePower：用户邀请算力

ownerPledgePower：用户质押算力

upper1：上 1 级地址

upper1InvitePower：上 1 级邀请算力

upper2：上 2 级地址

upper2InvitePower：上 2 级邀请算力

*/

event UpdatePower(uint256 pool, address lp, uint256 totalPower, address indexed owner, uint256 ownerInvitePower, uint256 ownerPledgePower, address indexed upper1, uint256 upper1InvitePower, address indexed upper2, uint256 upper2InvitePower);

//算力

/**

解质押

pool：矿池序号

lp：lp 合约地址

to：解质押转入地址

amount：解质押数量

*/

event UnStake(uint256 pool, address indexed lp, address indexed to, uint256 amount);

/**

提取奖励

pool：矿池序号

lp：lp 合约地址

to：奖励转入地址

amount：奖励数量

*/

event WithdrawReward(uint256 pool, address indexed lp, address indexed to, uint256 amount);

/**

挖矿

pool：矿池序号

lp：lp 合约地址

amount：奖励数量

*/

event Mint(uint256 pool, address indexed lp, uint256 amount);

//

/**

修改 OWNER

*/

function transferOwnership(address) external;

/**

设置 YOU

*/

function setYou(ITokenYou) external;

/**

设置邀请关系

*/

function setInvite(YouswapInviteV1) external;

/**

质押

*/

function deposit(uint256, uint256) external;

/**

解质押、提取奖励

*/

function withdraw(uint256, uint256) external;

/**

```
矿池质押地址
 */
function poolPledgeAddresss(uint256) external view returns (address[] memory);

/**
算力占比
 */
function powerScale(uint256, address) external view returns (uint256);

/**
待领取的奖励
 */
function pendingReward(uint256, address) external view returns (uint256);

/**
下级收益贡献
 */
function rewardContribute(address, address) external view returns (uint256);

/**
个人收益加成
 */
function selfReward(address) external view returns (uint256);

/**
通过 lp 查询矿池编号
 */
function poolNumbers(address) external view returns (uint256[] memory);

/**
设置运营权限
 */
function setOperateOwner(address, bool) external;
```

```
////////////////////////////////////////////////////////////////////////////////

/**
新建矿池
 */
function addPool(string memory, address, uint256, uint256) external returns (bool);

/**
修改矿池区块奖励
 */
function setRewardPerBlock(uint256, uint256) external;

/**
修改矿池总奖励
 */
function setRewardTotal(uint256, uint256) external;

/**
修改矿池名称
 */
function setName(uint256, string memory) external;

/**
修改矿池倍数
 */
function setMultiple(uint256, uint256) external;

/**
修改矿池排序
 */
function setPriority(uint256, uint256) external;
```

```
    ////////////////////////////////////////////////////////////////////////////////

}
// File: localhost/contract/implement/YouswapFactoryV1.sol



pragma solidity 0.7.4;









contract YouswapFactoryV1 is IYouswapFactoryV1 {

    using SafeMath for uint256;
    using SafeERC20 for IERC20;

    uint256 public deployBlock;//合约部署块高
    address public owner;//所有权限
    mapping(address => bool) public operateOwner;//运营权限
    ITokenYou public you;//you contract
    YouswapInviteV1 public invite;//invite contract

    uint256 public poolCount = 0;//矿池数量
    mapping(address => RewardInfo) public rewardInfos;//用户挖矿信息
    mapping(uint256 => PoolInfo) public poolInfos;//矿池信息
    mapping(uint256 => PoolViewInfo) public poolViewInfos;//矿池信息
    mapping(uint256 => address[]) public pledgeAddresss;//矿池质押地址
    mapping(uint256 => mapping(address => UserInfo)) public pledgeUserInfo;//矿池质押用户信息

    uint256 public constant inviteSelfReward = 5;//质押自奖励，5%
    uint256 public constant invite1Reward = 15;//1 级邀请奖励，15%
```

```
uint256 public constant invite2Reward = 10;//2 级邀请奖励，10%
uint256 public constant rewardPerBlock = 267094;//区块奖励
uint256 public rewardTotal = 0;//总挖矿奖励

constructor (ITokenYou _you, YouswapInviteV1 _invite) {
    deployBlock = block.number;
    owner = msg.sender;
    invite = _invite;
    _setOperateOwner(owner, true);
    _setYou(_you);
}

////////////////////////////////////////////////////////////////////////////////

function transferOwnership(address _owner) override external {
    require(owner == msg.sender, ErrorCode.FORBIDDEN);
    require((address(0) != _owner) && (owner != _owner), ErrorCode.INVALID_ADDRESSES);
    _setOperateOwner(owner, false);
    _setOperateOwner(_owner, true);
    owner = _owner;
}

function setYou(ITokenYou _you) override external {
    _setYou(_you);
}

function _setYou(ITokenYou _you) internal {
    require(owner == msg.sender, ErrorCode.FORBIDDEN);
    you = _you;
}

function setInvite(YouswapInviteV1 _invite) override external {
    require(owner == msg.sender, ErrorCode.FORBIDDEN);
```

```
        invite = _invite;
    }

    function deposit(uint256 _pool, uint256 _amount) override external {
        require(0 < _amount, ErrorCode.FORBIDDEN);
        PoolInfo storage poolInfo = poolInfos[_pool];
        require((address(0) != poolInfo.lp) && (poolInfo.startBlock <= block.number),
ErrorCode.MINING_NOT_STARTED);
        //require(0 == poolInfo.endBlock, ErrorCode.END_OF_MINING);
        (, uint256 startBlock) = invite.inviteUserInfoV2(msg.sender);
        if (0 == startBlock) {
            invite.register();

            emit InviteRegister(msg.sender);
        }

        IERC20(poolInfo.lp).safeTransferFrom(msg.sender, address(this), _amount);

        (address upper1, address upper2) = invite.inviteUpper2(msg.sender);

        computeReward(_pool);

        provideReward(_pool, poolInfo.rewardPerShare, poolInfo.lp, msg.sender, upper1, upper2);

        addPower(_pool, msg.sender, _amount, upper1, upper2);

        setRewardDebt(_pool, poolInfo.rewardPerShare, msg.sender, upper1, upper2);

        emit Stake(_pool, poolInfo.lp, msg.sender, _amount);
    }

    function withdraw(uint256 _pool, uint256 _amount) override external {
        PoolInfo storage poolInfo = poolInfos[_pool];
```

```
        require((address(0) != poolInfo.lp) && (poolInfo.startBlock <= block.number),
ErrorCode.MINING_NOT_STARTED);
        if (0 < _amount) {
            UserInfo storage userInfo = pledgeUserInfo[_pool][msg.sender];
            require(_amount <= userInfo.amount, ErrorCode.BALANCE_INSUFFICIENT);
            IERC20(poolInfo.lp).safeTransfer(msg.sender, _amount);

            emit UnStake(_pool, poolInfo.lp, msg.sender, _amount);
        }

        (address _upper1, address _upper2) = invite.inviteUpper2(msg.sender);

        computeReward(_pool);

        provideReward(_pool, poolInfo.rewardPerShare, poolInfo.lp, msg.sender, _upper1, _upper2);

        if (0 < _amount) {
            subPower(_pool, msg.sender, _amount, _upper1, _upper2);
        }

        setRewardDebt(_pool, poolInfo.rewardPerShare, msg.sender, _upper1, _upper2);
    }

    function poolPledgeAddresss(uint256 _pool) override external view returns (address[] memory) {
        return pledgeAddresss[_pool];
    }

    function computeReward(uint256 _pool) internal {
        PoolInfo storage poolInfo = poolInfos[_pool];
        if ((0 < poolInfo.totalPower) && (poolInfo.rewardProvide < poolInfo.rewardTotal)) {
            uint256 reward = (block.number -
poolInfo.lastRewardBlock).mul(poolInfo.rewardPerBlock);
            if (poolInfo.rewardProvide.add(reward) > poolInfo.rewardTotal) {
```

```
                reward = poolInfo.rewardTotal.sub(poolInfo.rewardProvide);
                poolInfo.endBlock = block.number;
            }

            rewardTotal = rewardTotal.add(reward);
            poolInfo.rewardProvide = poolInfo.rewardProvide.add(reward);
            poolInfo.rewardPerShare =
poolInfo.rewardPerShare.add(reward.mul(1e24).div(poolInfo.totalPower));
            poolInfo.lastRewardBlock = block.number;

            emit Mint(_pool, poolInfo.lp, reward);

            if (0 < poolInfo.endBlock) {
                emit EndPool(_pool, poolInfo.lp);
            }
        }
    }

    function addPower(uint256 _pool, address _user, uint256 _amount, address _upper1, address
_upper2) internal {
        PoolInfo storage poolInfo = poolInfos[_pool];
        poolInfo.amount = poolInfo.amount.add(_amount);

        uint256 pledgePower = _amount;
        UserInfo storage userInfo = pledgeUserInfo[_pool][_user];
        userInfo.amount = userInfo.amount.add(_amount);
        userInfo.pledgePower = userInfo.pledgePower.add(pledgePower);
        poolInfo.totalPower = poolInfo.totalPower.add(pledgePower);
        if (0 == userInfo.startBlock) {
            userInfo.startBlock = block.number;
            pledgeAddresss[_pool].push(msg.sender);
        }
```

```
        uint256 upper1InvitePower = 0;
        uint256 upper2InvitePower = 0;

        if (address(0) != _upper1) {
            uint256 inviteSelfPower = pledgePower.mul(inviteSelfReward).div(100);
            userInfo.invitePower = userInfo.invitePower.add(inviteSelfPower);
            poolInfo.totalPower = poolInfo.totalPower.add(inviteSelfPower);

            uint256 invite1Power = pledgePower.mul(invite1Reward).div(100);
            UserInfo storage upper1Info = pledgeUserInfo[_pool][_upper1];
            upper1Info.invitePower = upper1Info.invitePower.add(invite1Power);
            upper1InvitePower = upper1Info.invitePower;
            poolInfo.totalPower = poolInfo.totalPower.add(invite1Power);
            if (0 == upper1Info.startBlock) {
                upper1Info.startBlock = block.number;
                pledgeAddresss[_pool].push(_upper1);
            }
        }

        if (address(0) != _upper2) {
            uint256 invite2Power = pledgePower.mul(invite2Reward).div(100);
            UserInfo storage upper2Info = pledgeUserInfo[_pool][_upper2];
            upper2Info.invitePower = upper2Info.invitePower.add(invite2Power);
            upper2InvitePower = upper2Info.invitePower;
            poolInfo.totalPower = poolInfo.totalPower.add(invite2Power);
            if (0 == upper2Info.startBlock) {
                upper2Info.startBlock = block.number;
                pledgeAddresss[_pool].push(_upper2);
            }
        }

        emit UpdatePower(_pool, poolInfo.lp, poolInfo.totalPower, _user, userInfo.invitePower,
userInfo.pledgePower, _upper1, upper1InvitePower, _upper2, upper2InvitePower);
```

```
    }

    function subPower(uint256 _pool, address _user, uint256 _amount, address _upper1, address _upper2) internal {
        PoolInfo storage poolInfo = poolInfos[_pool];
        UserInfo storage userInfo = pledgeUserInfo[_pool][_user];
        poolInfo.amount = poolInfo.amount.sub(_amount);

        uint256 pledgePower = _amount;
        userInfo.amount = userInfo.amount.sub(_amount);
        userInfo.pledgePower = userInfo.pledgePower.sub(pledgePower);
        poolInfo.totalPower = poolInfo.totalPower.sub(pledgePower);

        uint256 upper1InvitePower = 0;
        uint256 upper2InvitePower = 0;

        if (address(0) != _upper1) {
            uint256 inviteSelfPower = pledgePower.mul(inviteSelfReward).div(100);
            userInfo.invitePower = userInfo.invitePower.sub(inviteSelfPower);
            poolInfo.totalPower = poolInfo.totalPower.sub(inviteSelfPower);

            UserInfo storage upper1Info = pledgeUserInfo[_pool][_upper1];
            if (0 < upper1Info.startBlock) {
                uint256 invite1Power = pledgePower.mul(invite1Reward).div(100);
                upper1Info.invitePower = upper1Info.invitePower.sub(invite1Power);
                upper1InvitePower = upper1Info.invitePower;
                poolInfo.totalPower = poolInfo.totalPower.sub(invite1Power);

                if (address(0) != _upper2) {
                    UserInfo storage upper2Info = pledgeUserInfo[_pool][_upper2];
                    if (0 < upper2Info.startBlock) {
                        uint256 invite2Power = pledgePower.mul(invite2Reward).div(100);
                        upper2Info.invitePower = upper2Info.invitePower.sub(invite2Power);
```

```
                upper2InvitePower = upper2Info.invitePower;
                poolInfo.totalPower = poolInfo.totalPower.sub(invite2Power);
            }
        }
    }
}

emit UpdatePower(_pool, poolInfo.lp, poolInfo.totalPower, _user, userInfo.invitePower, userInfo.pledgePower, _upper1, upper1InvitePower, _upper2, upper2InvitePower);
}

function provideReward(uint256 _pool, uint256 _rewardPerShare, address _lp, address _user, address _upper1, address _upper2) internal {
    uint256 inviteReward = 0;
    uint256 pledgeReward = 0;
    UserInfo storage userInfo = pledgeUserInfo[_pool][_user];
    if ((0 < userInfo.invitePower) || (0 < userInfo.pledgePower)) {
        inviteReward = userInfo.invitePower.mul(_rewardPerShare).sub(userInfo.inviteRewardDebt).div(1e24);
        pledgeReward = userInfo.pledgePower.mul(_rewardPerShare).sub(userInfo.pledgeRewardDebt).div(1e24);

        userInfo.pendingReward = userInfo.pendingReward.add(inviteReward.add(pledgeReward));

        RewardInfo storage userRewardInfo = rewardInfos[_user];
        userRewardInfo.inviteReward = userRewardInfo.inviteReward.add(inviteReward);
        userRewardInfo.pledgeReward = userRewardInfo.pledgeReward.add(pledgeReward);
    }

    if (0 < userInfo.pendingReward) {
        you.mint(_user, userInfo.pendingReward);
```

```
                RewardInfo storage userRewardInfo = rewardInfos[_user];
                userRewardInfo.receiveReward = userRewardInfo.inviteReward;

                emit WithdrawReward(_pool, _lp, _user, userInfo.pendingReward);

                userInfo.pendingReward = 0;
            }

            if (address(0) != _upper1) {
                UserInfo storage upper1Info = pledgeUserInfo[_pool][_upper1];
                if ((0 < upper1Info.invitePower) || (0 < upper1Info.pledgePower)) {
                    inviteReward =
upper1Info.invitePower.mul(_rewardPerShare).sub(upper1Info.inviteRewardDebt).div(1e24);
                    pledgeReward =
upper1Info.pledgePower.mul(_rewardPerShare).sub(upper1Info.pledgeRewardDebt).div(1e24);

                    upper1Info.pendingReward =
upper1Info.pendingReward.add(inviteReward.add(pledgeReward));

                    RewardInfo storage upper1RewardInfo = rewardInfos[_upper1];
                    upper1RewardInfo.inviteReward =
upper1RewardInfo.inviteReward.add(inviteReward);
                    upper1RewardInfo.pledgeReward =
upper1RewardInfo.pledgeReward.add(pledgeReward);
                }

                if (address(0) != _upper2) {
                    UserInfo storage upper2Info = pledgeUserInfo[_pool][_upper2];
                    if ((0 < upper2Info.invitePower) || (0 < upper2Info.pledgePower)) {
                        inviteReward =
upper2Info.invitePower.mul(_rewardPerShare).sub(upper2Info.inviteRewardDebt).div(1e24);
                        pledgeReward =
upper2Info.pledgePower.mul(_rewardPerShare).sub(upper2Info.pledgeRewardDebt).div(1e24);
```

```
                upper2Info.pendingReward = upper2Info.pendingReward.add(inviteReward.add(pledgeReward));

                RewardInfo storage upper2RewardInfo = rewardInfos[_upper2];
                upper2RewardInfo.inviteReward = upper2RewardInfo.inviteReward.add(inviteReward);
                upper2RewardInfo.pledgeReward = upper2RewardInfo.pledgeReward.add(pledgeReward);
            }
        }
    }
}

function setRewardDebt(uint256 _pool, uint256 _rewardPerShare, address _user, address _upper1, address _upper2) internal {
    UserInfo storage userInfo = pledgeUserInfo[_pool][_user];
    userInfo.inviteRewardDebt = userInfo.invitePower.mul(_rewardPerShare);
    userInfo.pledgeRewardDebt = userInfo.pledgePower.mul(_rewardPerShare);

    if (address(0) != _upper1) {
        UserInfo storage upper1Info = pledgeUserInfo[_pool][_upper1];
        upper1Info.inviteRewardDebt = upper1Info.invitePower.mul(_rewardPerShare);
        upper1Info.pledgeRewardDebt = upper1Info.pledgePower.mul(_rewardPerShare);

        if (address(0) != _upper2) {
            UserInfo storage upper2Info = pledgeUserInfo[_pool][_upper2];
            upper2Info.inviteRewardDebt = upper2Info.invitePower.mul(_rewardPerShare);
            upper2Info.pledgeRewardDebt = upper2Info.pledgePower.mul(_rewardPerShare);
        }
    }
}
```

```
    function powerScale(uint256 _pool, address _user) override external view returns (uint256) {
        PoolInfo memory poolInfo = poolInfos[_pool];
        if (0 == poolInfo.totalPower) {
            return 0;
        }

        UserInfo memory userInfo = pledgeUserInfo[_pool][_user];
        return (userInfo.invitePower.add(userInfo.pledgePower).mul(100)).div(poolInfo.totalPower);
    }

    function pendingReward(uint256 _pool, address _user) override external view returns (uint256) {
        uint256 totalReward = 0;
        PoolInfo memory poolInfo = poolInfos[_pool];
        if (address(0) != poolInfo.lp && (poolInfo.startBlock <= block.number)) {
            uint256 rewardPerShare = 0;
            if (0 < poolInfo.totalPower) {
                uint256 reward = (block.number - poolInfo.lastRewardBlock).mul(poolInfo.rewardPerBlock);
                if (poolInfo.rewardProvide.add(reward) > poolInfo.rewardTotal) {
                    reward = poolInfo.rewardTotal.sub(poolInfo.rewardProvide);
                }
                rewardPerShare = reward.mul(1e24).div(poolInfo.totalPower);
            }
            rewardPerShare = rewardPerShare.add(poolInfo.rewardPerShare);

            UserInfo memory userInfo = pledgeUserInfo[_pool][_user];
            totalReward = userInfo.pendingReward;
            totalReward = totalReward.add(userInfo.invitePower.mul(rewardPerShare).sub(userInfo.inviteRewardDebt).div(1e24));
            totalReward = totalReward.add(userInfo.pledgePower.mul(rewardPerShare).sub(userInfo.pledgeRewardDebt).div(1e24));
```

```
        }

        return totalReward;
    }

    function rewardContribute(address _user, address _lower) override external view returns (uint256)
{
        if ((address(0) == _user) || (address(0) == _lower)) {
            return 0;
        }

        uint256 inviteReward = 0;
        (address upper1, address upper2) = invite.inviteUpper2(_lower);
        if (_user == upper1) {
            inviteReward = rewardInfos[_lower].pledgeReward.mul(invite1Reward).div(100);
        }else if (_user == upper2) {
            inviteReward = rewardInfos[_lower].pledgeReward.mul(invite2Reward).div(100);
        }

        return inviteReward;
    }

    function selfReward(address _user) override external view returns (uint256) {
        address upper1 = invite.inviteUpper1(_user);
        if (address(0) == upper1) {
            return 0;
        }

        RewardInfo memory userRewardInfo = rewardInfos[_user];
        return userRewardInfo.pledgeReward.mul(inviteSelfReward).div(100);
    }

    function poolNumbers(address _lp) override external view returns (uint256[] memory) {
```

```
        uint256 count = 0;
        for (uint256 i = 0; i < poolCount; i++) {
            if (_lp == poolViewInfos[i].lp) {
                count = count.add(1);
            }
        }

        uint256[] memory numbers = new uint256[](count);
        count = 0;
        for (uint256 i = 0; i < poolCount; i++) {
            if (_lp == poolViewInfos[i].lp) {
                numbers[count] = i;
                count = count.add(1);
            }
        }

        return numbers;
    }

    function setOperateOwner(address _address, bool _bool) override external {
        _setOperateOwner(_address, _bool);
    }

    function _setOperateOwner(address _address, bool _bool) internal {
        require(owner == msg.sender, ErrorCode.FORBIDDEN);
        operateOwner[_address] = _bool;
    }

    ////////////////////////////////////////////////////////////////////////////////

    function addPool(string memory _name, address _lp, uint256 _startBlock, uint256 _rewardTotal)
override external returns (bool) {
        require(operateOwner[msg.sender] && (address(0) != _lp) && (address(this) != _lp),
```

```
ErrorCode.FORBIDDEN);
        _startBlock = _startBlock < block.number ? block.number : _startBlock;
        uint256 _pool = poolCount;
        poolCount = poolCount.add(1);

        PoolViewInfo storage poolViewInfo = poolViewInfos[_pool];
        poolViewInfo.lp = _lp;
        poolViewInfo.name = _name;
        poolViewInfo.multiple = 1;
        poolViewInfo.priority = _pool.mul(100);

        PoolInfo storage poolInfo = poolInfos[_pool];
        poolInfo.startBlock = _startBlock;
        poolInfo.rewardTotal = _rewardTotal;
        poolInfo.rewardProvide = 0;
        poolInfo.lp = _lp;
        poolInfo.amount = 0;
        poolInfo.lastRewardBlock = _startBlock.sub(1);
        poolInfo.rewardPerBlock = rewardPerBlock;
        poolInfo.totalPower = 0;
        poolInfo.endBlock = 0;
        poolInfo.rewardPerShare = 0;

        emit UpdatePool(true, _pool, poolInfo.lp, poolViewInfo.name, poolInfo.startBlock,
poolInfo.rewardTotal, poolInfo.rewardPerBlock, poolViewInfo.multiple, poolViewInfo.priority);

        return true;
    }

    function setRewardPerBlock(uint256 _pool, uint256 _rewardPerBlock) override external {
        require(operateOwner[msg.sender], ErrorCode.FORBIDDEN);
        PoolInfo storage poolInfo = poolInfos[_pool];
        require((address(0) != poolInfo.lp) && (0 == poolInfo.endBlock),
```

```
ErrorCode.POOL_NOT_EXIST_OR_END_OF_MINING);
        poolInfo.rewardPerBlock = _rewardPerBlock;

        PoolViewInfo memory poolViewInfo = poolViewInfos[_pool];

        emit UpdatePool(false, _pool, poolInfo.lp, poolViewInfo.name, poolInfo.startBlock,
poolInfo.rewardTotal, poolInfo.rewardPerBlock, poolViewInfo.multiple, poolViewInfo.priority);
    }

    function setRewardTotal(uint256 _pool, uint256 _rewardTotal) override external {
        require(operateOwner[msg.sender], ErrorCode.FORBIDDEN);
        PoolInfo storage poolInfo = poolInfos[_pool];
        require((address(0) != poolInfo.lp) && (0 == poolInfo.endBlock),
ErrorCode.POOL_NOT_EXIST_OR_END_OF_MINING);
        require(poolInfo.rewardProvide < _rewardTotal,
ErrorCode.REWARDTOTAL_LESS_THAN_REWARDPROVIDE);
        poolInfo.rewardTotal = _rewardTotal;

        PoolViewInfo memory poolViewInfo = poolViewInfos[_pool];

        emit UpdatePool(false, _pool, poolInfo.lp, poolViewInfo.name, poolInfo.startBlock,
poolInfo.rewardTotal, poolInfo.rewardPerBlock, poolViewInfo.multiple, poolViewInfo.priority);
    }

    function setName(uint256 _pool, string memory _name) override external {
        require(operateOwner[msg.sender], ErrorCode.FORBIDDEN);
        PoolViewInfo storage poolViewInfo = poolViewInfos[_pool];
        require(address(0) != poolViewInfo.lp,
ErrorCode.POOL_NOT_EXIST_OR_END_OF_MINING);
        poolViewInfo.name = _name;

        PoolInfo memory poolInfo = poolInfos[_pool];
```

```
        emit UpdatePool(false, _pool, poolInfo.lp, poolViewInfo.name, poolInfo.startBlock, poolInfo.rewardTotal, poolInfo.rewardPerBlock, poolViewInfo.multiple, poolViewInfo.priority);
    }

    function setMultiple(uint256 _pool, uint256 _multiple) override external {
        require(operateOwner[msg.sender], ErrorCode.FORBIDDEN);
        PoolViewInfo storage poolViewInfo = poolViewInfos[_pool];
        require(address(0) != poolViewInfo.lp, ErrorCode.POOL_NOT_EXIST_OR_END_OF_MINING);
        poolViewInfo.multiple = _multiple;

        PoolInfo memory poolInfo = poolInfos[_pool];

        emit UpdatePool(false, _pool, poolInfo.lp, poolViewInfo.name, poolInfo.startBlock, poolInfo.rewardTotal, poolInfo.rewardPerBlock, poolViewInfo.multiple, poolViewInfo.priority);
    }

    function setPriority(uint256 _pool, uint256 _priority) override external {
        require(operateOwner[msg.sender], ErrorCode.FORBIDDEN);
        PoolViewInfo storage poolViewInfo = poolViewInfos[_pool];
        require(address(0) != poolViewInfo.lp, ErrorCode.POOL_NOT_EXIST_OR_END_OF_MINING);
        poolViewInfo.priority = _priority;

        PoolInfo memory poolInfo = poolInfos[_pool];

        emit UpdatePool(false, _pool, poolInfo.lp, poolViewInfo.name, poolInfo.startBlock, poolInfo.rewardTotal, poolInfo.rewardPerBlock, poolViewInfo.multiple, poolViewInfo.priority);
    }

    ////////////////////////////////////////////////////////////////////////////////

}
```

6. 附录 B：安全风险评级标准

智能合约漏洞评级标准	
漏洞评级	**漏洞评级说明**
高危漏洞	能直接造成代币合约或用户资金损失的漏洞，如：能造成代币价值归零的数值溢出漏洞、能造成交易所损失代币的假充值漏洞、能造成合约账户损失 ETH 或代币的重入漏洞等； 能造成代币合约归属权丢失的漏洞，如：关键函数的访问控制缺陷、call 注入导致关键函数访问控制绕过等； 能造成代币合约无法正常工作的漏洞，如：因向恶意地址发送 ETH 导致的拒绝服务漏洞、因 gas 耗尽导致的拒绝服务漏洞。
中危漏洞	需要特定地址才能触发的高风险漏洞，如代币合约拥有者才能触发的数值溢出漏洞等；非关键函数的访问控制缺陷、不能造成直接资金损失的逻辑设计缺陷等。
低危漏洞	难以被触发的漏洞、触发之后危害有限的漏洞，如需要大量 ETH 或代币才能触发的数值溢出漏洞、触发数值溢出后攻击者无法直接获利的漏洞、通过指定高 gas 触发的事务顺序依赖风险等。

7. 附录 C：智能合约安全审计工具简介

7.1 Manticore

Manticore 是一个分析二进制文件和智能合约的符号执行工具，Manticore 包含一个符号虚拟机（EVM），一个 EVM 反汇编器/汇编器以及一个用于自动编译和分析 Solidity 的方便界面。它还集成了 Ethersplay，用于 EVM 字节码的 Bit of Traits of Bits 可视化反汇编程序，用于可视化分析。与二进制文件一样，Manticore 提供了一个简单的命令行界面和一个用于分析 EVM 字节码的 Python API。

7.2 Oyente

Oyente 是一个智能合约分析工具，Oyente 可以用来检测智能合约中常见的 bug，比如 reentrancy、事务排序依赖等等。更方便的是，Oyente 的设计是模块化的，所以这让高级用户可以实现并插入他们自己的检测逻辑，以检查他们的合约中自定义的属性。

7.3 securify.sh

Securify 可以验证智能合约常见的安全问题，例如交易乱序和缺少输入验证，它在全自动化的同时分析程序所有可能的执行路径，此外，Securify 还具有用于指定漏洞的特定语言，这使 Securify 能够随时关注当前的安全性和其他可靠性问题。

7.4 Echidna

Echidna 是一个为了对 EVM 代码进行模糊测试而设计的 Haskell 库。

7.5 MAIAN

MAIAN 是一个用于查找智能合约漏洞的自动化工具，Maian 处理合约的字

节码，并尝试建立一系列交易以找出并确认错误。

7.6 ethersplay

ethersplay 是一个 EVM 反汇编器，其中包含了相关分析工具。

7.7 ida-evm

ida-evm 是一个针对虚拟机（EVM）的 IDA 处理器模块。

7.8 Remix-ide

Remix 是一款基于浏览器的编译器和 IDE，可让用户使用 Solidity 语言构建合约并调试交易。

7.9 知道创宇区块链安全审计人员专用工具包

知道创宇安全审计人员专用工具包，由知道创宇渗透测试工程师研发，收集和使用，包含专用于测试人员的批量自动测试工具，自主研发的工具、脚本或利用工具等。

知道创宇

北京知道创宇信息技术股份有限公司

咨询电话 +86(10)400 060 9587

邮　　箱 sec@knownsec.com

官　　网 www.knownsec.com

地　　址 北京市 朝阳区 望京 SOHO T2-B座-2509